达梦数据库应用基础

主　　编　曾昭文　龚建华

副 主 编　付　铨　冯勤群　张　胜

编撰人员（按姓氏笔画排序）

王　强　文　峰　冯勤群　左青云

付　铨　朱明东　刘志红　吴照林

张　胜　张海粟　周英彪　徐　飞

龚建华　曾昭文　戴剑伟　薛　慧

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书以达梦数据库管理系统 DM 7.1 为蓝本，全面系统地介绍了达梦数据库体系结构、数据库日常维护操作和数据库基本参数设置，是学习达梦数据库的基础教材和参考用书。

全书共 7 章，主要包括达梦数据库概述、安装与卸载、表空间管理、对象管理、备份与还原、作业管理、安全管理等内容。对数据库的各项管理工作，本书中都列举了详细的例子，既介绍了 SQL 命令方式的管理方法，又介绍了可视化图形界面的管理方法，适合不同学习进度的读者使用。附光盘 1 张，内含达梦数据库管理系统 7.1 标准版及例题源码。

本书内容全面、举例丰富、操作性强，语言通俗、格式规范，可作为相关专业本科生的教材，也可作为工程技术人员的参考用书。

图书在版编目（CIP）数据

达梦数据库应用基础 / 曾昭文，龚建华主编. —北京：电子工业出版社，2016.11
ISBN 978-7-121-30212-1

Ⅰ. ①达… Ⅱ. ①曾… ②龚… Ⅲ. ①关系数据库系统 Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字（2016）第 257856 号

策划编辑：李 敏
责任编辑：郝黎明
印 刷：
装 订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：15 字数：351 千字
版 次：2016 年 11 月第 1 版
印 次：2016 年 11 月第 1 次印刷
定 价：49.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-88254753 或 limin@phei.com.cn。

前　言

发展具有自主知识产权的国产数据库管理系统，打破国外数据库产品的垄断，为我国信息化建设提供安全可控的基础软件，是维护国家信息安全的重要手段。

达梦数据库管理系统作为国内最早推出的具有自主知识产权的数据库管理系统之一，是唯一获得国家自主原创产品认证的数据库产品，现已在公安、电力、铁路、航空、审计、通信、金融、海关、国土资源、电子政务等多个领域得到广泛应用，为国家机关、各级政府和企业信息化建设发挥了积极作用。

为了推动国产数据库管理系统的教学和人才培养，促进国产数据库的广泛应用，我们在总结数据库管理系统长期教学和科研实践经验的基础上，在达梦数据库有限公司的大力支持下，以达梦数据库 DM7.1 为蓝本，编写了《达梦数据库应用基础》和《达梦数据库 SQL 指南》两本教材。

《达梦数据库应用基础》全面系统地介绍了达梦数据库的体系结构、基本参数和数据库的日常维护操作。全书共 7 章，包括达梦数据库概述、安装与卸载、表空间管理、对象管理、备份与还原、作业管理、安全管理，以及附录等。对数据库各项管理工作，书中都列举了详细的例子，既介绍了 SQL 命令方式的管理方法，又介绍了可视化图形界面的管理方法，适合不同学习基础的读者使用。

吴照林确定了本书的编写定位和要求。大纲由曾昭文、龚建华拟制，第 1 章和附录由曾昭文执笔，第 2 章由张胜执笔，第 3、4、5、6 章由龚建华执笔，第 7 章由戴剑伟执笔，付铨、刘志红、文峰、朱明东、冯勤群等同志在本书编写过程中承担了大量工作，最后统稿修改由曾昭文、龚建华完成。

在本书的编写过程中，编者参考了达梦数据库有限公司提供的技术资料，在此表示衷心的感谢。

由于编者水平有限，加之时间仓促，书中难免有错误与不妥之处，敬请读者批评指正，欢迎读者通过电子邮件 gongjh123@163.com 与我们交流。

编　者

2016 年 10 月于武汉

目　　录

1

第 1 章

达梦数据库概述

达梦数据库（简称 DM）是达梦数据库有限公司推出的具有完全自主知识产权的大型通用关系型数据库管理系统，是在总结 DM 系列产品研发与应用经验的基础之上，吸收主流数据库产品的优点，采用类 Java 的虚拟机技术设计的新一代数据库产品。

1.1 DM 7 主要特性

DM 7 采用全新的体系架构，在保证大型通用的基础上，针对可靠性、高性能、海量数据处理和安全性做了大量的研发和改进工作，极大地提升了达梦数据库产品的性能、语言的丰富性、可扩展性，能同时兼顾 OLTP 和 OLAP 请求，从根本上提升了 DM 7 产品的品质。

1.1.1 通用性

DM 7 产品的通用性主要体现在以下几个方面。

1．硬件平台支持

DM 7 兼容多种硬件体系，可运行于 X86、SPARC、Power 等硬件体系之上。DM 7 各种平台上的数据存储结构和消息通信结构完全一致，使得 DM 7 各种组件在不同的硬件平台上具有一致的使用特性。

2．操作系统支持

DM 7 实现了平台无关性，支持 Windows 系列、Linux（2.4 及 2.4 以上内核）、UNIX、

Kylin、AIX、Solaris 等主流操作系统。DM 7 的服务器、接口程序和管理工具均可在 32 位/64 位版本操作系统上使用。

3．应用开发支持

1）开发环境支持

DM 7 支持多种主流集成开发环境，包括 PowerBuilder、Delphi、Visual Studio、.NET、C++Builder、Qt、JBuilder、Eclipse、Zend Studio 等。

2）开发框架技术支持

DM 7 支持各种开发框架技术，主要有 Spring、Hibernate、iBATIS SQL Map、Entity Framework、Zend Framework 等。

3）中间件支持

DM 7 支持主流系统中间件，包括 WebLogic、WebSphere、Tomcat、Jboss、东方通 TongWeb、金蝶 Apusic、中创 InfoWeb 等。

4．标准接口支持

DM 7 提供对 SQL92 的特性支持以及 SQL99 的核心级别支持；支持多种数据库开发接口，包括 OLE DB、ADO、ODBC、OCI、JDBC、Hibernate、PHP、PDO、DB Express 以及.NET DataProvider 等。

5．网络协议支持

DM 7 支持多种网络协议，包括 IPv4 协议、IPv6 协议等。

6．字符集支持

DM 7 完全支持 Unicode、GBK18030 等常用字符集。

7．国际化支持

DM 7 提供了国际化支持，服务器和客户端工具均支持简体中文和英文来显示输出结果和错误信息。

1.1.2 高可用性

1．快速的自动故障恢复

DM 7 通过 REDO 日志记录数据库的物理文件变化信息。当发生系统故障的时候（如机器掉电），系统通过 REDO 日志进行重做处理，恢复用户的数据和回滚信息，从而使数据库系统从故障中恢复，避免数据丢失，确保事务的完整性。相对达梦以前的版本，DM 7 改进了 REDO 日志的管理策略。采用逻辑 LSN 值替代了原有的物理文件地址映射到 LSN 生成机制，极大简化了 REDO 日志的处理逻辑。

REDO 日志支持压缩存储，可以减少存储空间开销。DM 7 在故障恢复时采用了并行处理机制执行 REDO 日志，有效减少了重做花费的时间。

2．逻辑日志

DM 7 在物理的 REDO 日志之外，又添加了逻辑日志。逻辑日志记录数据库表上的所有插入、删除、更新等数据变化。可以指定部分表记录逻辑日志，也可以设置所有表都记录逻辑日志。借助逻辑日志，DM 7 可以提供操作分析、数据重演以及数据复制等高级功能。

3．可靠的备份与还原

DM 7 可以提供数据库或整个服务器的冷/热备份以及对应的还原功能，达到数据库数据的保护和迁移。DM 7 支持的备份类型包括物理备份、逻辑备份和 B 树备份，其中 B 树备份是介于物理备份和逻辑备份之间的一种形态。

DM 7 支持增量备份，支持 LSN 和时间点还原；可备份不同级别的数据，包括数据库级、表空间级和表级；支持在联机、脱机的状态下进行备份、还原操作。

4．高级复制

DM 7 的复制功能基于逻辑日志实现。主机将逻辑日志发往从机，而从机根据日志模拟事务与语句重复主机的数据操作。相对语句级的复制，逻辑日志可以更准确地反映主机数据的时序变化，从而减少冲突，提高数据复制的一致性。

DM 7 提供基于事务的同步复制和异步复制功能。同步复制即所有复制节点的数据是同步的，如果复制环境中的主表数据发生了变化，这种改变将以事务为单位同步传播和应用到其他所有复制节点。异步复制是指在多个复制节点之间，主节点的数据更新需要经过一定的时间周期之后才反映到从节点。如果复制环境中主节点要被复制的数据发生了更新操作，这种改变将在不同的事务中被传播和应用到其他所有从节点。这些不同的事务间可以间隔几秒、几分钟、几小时，也可以是几天之后。复制节点之间的数据在一段时间内是不同步的，但传播最终将保证所有复制节点间的数据一致。数据复制功能支持一到多、多到一、级联复制、多主多从复制、环形复制、对称复制以及大数据对象复制。

5．基于 REDO 日志的主备系统——数据守护

主备系统是 DM 7 提高容灾能力的重要手段。系统由一台主机与一或多台备机构成。主机提供正常的数据处理服务。备机则时刻保持与主机的数据同步。一旦主机发生故障，备机中的一台立刻可以切换成为新的主机，继续提供服务。主备机的切换是通过服务器、观察器与接口自动完成的，对客户端几乎完全透明。

DM 7 的主备系统基于优化后的 REDO 日志系统开发，其功能更加稳定可靠。主备机间传递压缩的日志数据，通信效率大大提升。DM 7 主备系统提供了配置模式，可在不停机状态下在单机系统与主备系统间平滑变换。

DM 7 的主备系统可提供全功能的数据库支持，客户端访问主机系统没有任何功能限制，而备机同样可以作为主机的只读镜像支持客户端的只读查询请求。

1.1.3 高性能

1．查询优化

DM 7 采用多趟扫描、代价估算的优化策略。系统基于数据字典信息、数据分布统计值、执行语句涉及的表、索引和分区的存储特点等统计信息实现了代价估算模型，在多个可行的执行计划中选择代价最小的作为最终执行计划。同时，DM 7 还支持查询计划的 HINT 功能，可供经验丰富的 DBA 对特定查询进行优化改进，进一步提高查询的效率和灵活性。

DM 7 查询优化器利用优化规则，将所有的相关子查询变换为等价的关系连接。相关子查询的平坦化，极大地降低了代价优化的算法复杂程度，使得优化器可以更容易地生成较优的查询计划。

2．虚拟机执行器

DM 7 实现了基于堆栈的虚拟机执行器。这种运行机制可以有效提升数据计算以及存储过程/函数的执行效率，具有以下特点：采用以字长为分配单位的标准堆栈，提高空间利用率，充分利用 CPU 的 2 级缓存，提升性能；增加栈帧概念，方便实现函数/方法的跳转，为 PL/SQL 脚本的调试提供了基础；采用内存运行堆的概念，实现对象、数组、动态的数据类型存储；采用面向栈的表达式计算模式，减少了虚拟机代码的体积、数据的移动；定义了指令系统，增加了对对象、方法、参数、堆栈的访问，便于 PL/SQL 的执行。

3．批量数据处理

当数据读入内存后，按照传统策略，需要经过逐行过滤、连接、计算等操作处理后，才能生成最终结果集。在海量的数据处理场景下，必然产生大量重复的函数调用及数据的反复复制与计算代价。

DM 7 引入了数据的批量处理技术，即读取一批，计算一批，传递一批，生成一批。数据批量处理具有显而易见的好处：内存紧靠在一起的数据执行批量计算，可以显著提升 Cache 命中率，从而提升内存处理效率；数据成批而非单行地抽取与传递，可以显著减少在上下层操作符间流转数据的函数调用次数；采用优化的引用方式在操作符间传递数据，可以有效降低数据复制的代价；系统标量函数支持批量计算，可以进一步减少函数调用次数。DM 7 采用批量数据处理策略，比一次一行的数据处理模式快 10～100 倍。

4．查询计划重用

SQL 语句从分析、优化到实际执行，每一步都需要消耗系统资源。查询计划的重用，可以减少重复分析操作，有效提升语句的执行效率。DM 7 采用参数化常量方法，使得常

量值不同的查询语句，同样可以重用查询计划。经此优化后的计划重用策略，在应用系统中的实用性明显增强了。

DM 7 采用“历史回溯”策略，对于数据的多版本并发控制实现了原生性支持。DM 7 改造了数据记录与回滚记录的结构。在数据记录中添加字段记录最近修改的事务 ID 及与其对应的回滚记录地址，而在回滚记录中也记录了该行上一更新操作的事务 ID 与相应回滚记录地址。通过数据记录与回滚记录的链接关系，构造出一行数据的完整更新历史。

5．查询结果集的缓存

DM 7 提供查询结果集缓存策略。相同的查询语句，如果涉及的表数据没有变化，则可以直接重用缓存的结果集。查询结果缓存，在数据变化不频繁的 OLAP 应用模式，或存在大量类似编目函数查询的应用环境下有非常良好的性能提升效果。

在服务器端实现结果集缓存，可以在提升查询速度的同时，保证缓存结果的实时性和正确性。

6．异步检查点技术

DM 7 采用更加有效的异步检查点机制。新检查点机制采用类似“蜻蜓点水”的策略，每次仅从缓冲区的更新链中摘取少量的更新页刷新。反复多次刷页达到设定的总数比例后，才相应调整检查点值。与原有检查点长时间占用缓冲区的策略相比，逻辑更加简单，速度更快，对整体系统运行影响更小。

7．多版本并发控制

DM 7 采用“历史回溯”策略，对于数据的多版本并发控制实现了原生性支持。DM 7 改造了数据记录与回滚记录的结构。在数据记录中添加字段记录最近修改的事务 ID 及与其对应的回滚记录地址，而在回滚记录中也记录了该行上一更新操作的事务 ID 与相应回滚记录地址。通过数据记录与回滚记录的链接关系，构造出一行数据的完整更新历史各版本。

DM 7 的多版本采用了并发控制技术，数据中仅存储最新一条记录，各个会话事务通过其对应可见事务集，利用回滚段记录组装出自己可见的版本数据。使用这种技术，不必保持冗余数据，也就避免了使用附加数据整理工具。多版本并发控制技术使得查询与更新操作间互不干扰，有效提高了高并发应用场景中的执行效率。

8．数据字典缓存技术

DM 7 中实现了数据字典缓存技术。DDL 语句被转换为基本的 DML 操作，执行期间不必封锁整个数据字典，可以有效降低 DDL 操作对整体系统并发执行的影响。在有较多 DDL 并发操作的系统中可有效提升系统性能。

9．可配置的工作线程模式

DM 7 的内核工作线程同时支持内核线程和用户态线程两种模式，通过配置参数即可

以实现两种模式的切换。

内核线程的切换完全由操作系统决定，但操作系统并不了解、也不关心应用逻辑，只能采取简单、通用的策略来平衡各个内核线程的 CPU 时间；在高并发情况下，往往导致很多无效的上下文切换，浪费了宝贵的 CPU 资源。用户态线程由用户指定线程切换策略，结合应用的实际情况，决定何时让出 CPU 的执行，可以有效避免过多的无效切换，提升系统性能。

DM 7 的工作线程在少量内核线程的基础上，模拟了大量的用户态线程（一般来说，工作线程数不超过 CPU 的核数，用户态线程数由数据库的连接数决定）。大量的用户态线程在内核线程内部自主调度，基本消除了由于操作系统调度产生的上下文切换；同时，由于内核线程数的减少，进一步降低了冲突产生的概率，有效提升了系统性能，特别是在高并发情况下的性能提升十分明显。

10．多缓冲区

DM 7 采用了多缓冲区机制，将数据缓冲区划成多个分片。数据页按照其页号，进入各自缓冲区分片。用户访问不同的缓冲区分片，不会导致访问冲突。高并发情况下，这种机制可以降低全局数据缓冲区的访问冲突。

DM 7 支持动态缓冲区管理，根据不同的系统资源情况，管理员可以配置缓冲区伸缩策略。

11．查询内并行处理

DM 7 为具有多个处理器（CPU）的计算机提供了并行查询，以优化查询执行和索引操作。并行查询的优势就是可以通过多个线程来处理查询作业，从而提高查询的效率。

在 DM 7 中有一个查询优化器，当对 SQL 语句进行优化后数据库才会去执行查询语句。

如果查询优化器认为查询语句可以从并行查询中获得较高效率，就会将本地通信操作符插入到查询执行计划中，为并行查询做准备。本地通信操作符是在查询执行计划中提供进程管理、数据重新分发和流控制的运算符。在查询计划执行过程中，数据库会确认当前的系统工作负荷和配置信息，判断是否有足够多的线程允许执行并行查询。确定最佳的线程数后，在查询计划初始化确定的线程上展开并行查询执行。在多个线程上并行执行查询时，查询将一直使用相同的线程数，直到完成。每次从高速缓存中检索查询执行计划时，DM 7 都重新检查最佳线程数。

12．分段式数据压缩

DM 7 支持数据压缩，即将一个字段的所有数据，分成多个小片压缩存储起来。系统采用智能压缩策略，根据采样值特征，自动选择最合适的压缩算法进行数据压缩。而多行相同类型数据一起压缩，可以显著提升数据的压缩比，进一步减少系统的空间资源开销。

13．行列融合

DM 7 同时支持行存储引擎与列存储引擎，可实现事务内对行存储表与列存储表的同

时访问，可同时适用于联机事务和分析处理。在并发量、数据量规模较小时，单机 DM 7 利用其行列融合特性，即可同时满足联机事务处理和联机分析处理的应用需求，并能够满足混合型的应用要求。

14．海量数据分析

DM 7 提供 OLAP 函数，用于支持复杂的分析操作，侧重对决策人员和高层管理人员的决策支持，可根据分析人员的要求快速、灵活地进行大数据量的复杂查询处理，并且以直观易懂的形式将查询结果提供给决策人员，以便他们准确掌握企业的经营状况，了解被服务对象的需求，制定正确的方案。

15．大规模并行处理架构

为了支持海量数据存储和处理、高并发处理、高性价比、高可用性等功能，提供高端数据仓库解决方案，DM 7 支持大规模并行处理 MPP 架构，以极低的成本代价，为客户提供业界领先的计算性能。DM 7 采用完全对等无共享的 MPP 架构，支持 SQL 并行处理，可自动化分区数据和并行查询，无 I/O 冲突。

DM 7 的 MPP 架构将负载分散到多个数据库服务器主机，实现了数据的分布式存储。采用了完全对等的无共享架构，每个数据库服务器称为一个 EP。这种架构中，节点没有主从之分，每个 EP 都能够对用户提供完整的数据库服务。在处理海量数据分析请求时，各个节点通过内部通信系统协同工作，通过并行运算技术大幅提高查询效率。

DM 7 MPP 为新一代数据仓库所需的大规模数据和复杂查询提供了先进的软件级解决方案，具有业界先进的架构和高度的可靠性，能帮助企业管理好数据，使之更好地服务于企业，推动数据依赖型企业的发展。

1.1.4　高安全性

1．安全等级

DM 7 是具有自主知识产权的高安全数据库管理系统，已通过公安部安全四级评测，是目前安全等级最高的商业数据库之一。同时，DM 7 通过了中国信息安全评测中心的 EAL3 级评测。

2．双因子结合的身份鉴别

DM 提供基于用户口令和用户数字证书相结合的用户身份鉴别功能。当接收的用户口令和用户数字证书均正确时，才算认证通过，若用户口令和用户数据证书有一个不正确或与相应的用户名不匹配，则认证不通过，这种增强的身份认证方式可以更好地防止口令被盗、冒充用户登录等情况，为数据库安全把好了第一道关。

另外，DM 7 还支持基于操作系统的身份认证、基于 LDAP 集中式的第三方认证。

3．审计分析与实时侵害检测

DM 7 提供数据库审计功能，审计类别包括：系统级审计、语句级审计、对象级审计。

DM 7 的审计记录存放在数据库外的专门审计文件中，保证审计数据的独立性。审计文件可以脱离数据库系统保存和复制，借助专用工具进行阅读、检索以及合并等维护操作。

DM 7 提供审计分析功能，通过审计分析工具 Analyzer 实现对审计记录的分析。用户能够根据所制定的分析规则，对审计记录进行分析，判断系统中是否存在对系统安全构成威胁的活动。

DM 7 提供强大的实时侵害检测功能，用于实时分析当前用户的操作，并查找与该操作相匹配的审计分析规则。根据规则判断用户行为是否为侵害行为，以及确定侵害等级，并根据侵害等级采取相应的响应措施。响应措施包括：实时报警生成、违例进程终止、服务取消和账号锁定或失效。

4．自主访问控制

DM 7 提供了系统权限和对象权限管理功能，并支持基于角色的权限管理，方便数据库管理员对用户访问权限进行灵活配置。

在 DM 7 中，可以对用户直接授权，也可以通过角色授权。角色表示一组权限的集合，数据库管理员可以通过创建角色来简化权限管理过程。可以把一些权限授予一个角色，而这个角色又可以被授予多个用户，从而使基于这些角色的用户间接地获得权限。在实际的权限分配方案中，通常先由数据库管理员为数据库定义一系列的角色，然后将权限分配给基于这些角色的用户。

5．所有主客体的强制访问控制

DM 7 提供强制访问控制功能，强制访问控制的范围涉及数据库内所有的主客体，该功能达到了安全四级的要求。强制访问控制是利用策略和标记实现数据库访问控制的一种机制。该功能主要针对数据库用户、各种数据库对象、表以及表内数据。控制粒度同时达到列级和记录级。

当用户操作数据库对象时，不仅要满足自主访问控制的权限要求，还要满足用户和数据之间标记的支配关系。这样就避免了管理权限全部由数据库管理员一人负责的局面，可以有效防止敏感信息的泄露与篡改，增强系统的安全性。

6．基于 SSL 协议的通信加密

DM 7 支持基于 SSL 协议的通信加密，对传输在客户端和服务器端的数据进行非对称的安全加密，保证数据在传输过程中的保密性、完整性、抗抵赖性。

7．存储加密

DM 7 实现了对存储数据的透明存储加密、半透明存储加密和非透明存储加密。每种模式均可自由配置加密算法。用户可以根据自己的需要自主选择采用何种加密模式。

8．资源限制

DM 7 实现了多种资源限制功能，包括并发会话总数、单用户会话数、用户会话 CPU 时间、用户请求 CPU 时间、会话读取页、请求读取页、会话私有内存等，这些资源限制项足够丰富并满足资源限制的要求，达到防止用户恶意抢占资源的目的，尽可能减少人为的安全隐患，保证所有数据库用户均能正常访问和操作数据库。DM 7 还可配置表的存储空间配额。

系统管理员可借此功能对每个数据库用户单独配置最合适的管理策略，并能有效防止各种恶意抢占资源的攻击。

9．加密引擎

DM 7 提供加密引擎功能，当 DM 7 内置的加密算法，如 AES 系列、DES 系列、DESEDE 系列、RC4 等加密算法，无法满足用户数据存储加密要求时，用户可能希望使用自己特殊的加密算法，或强度更高的加密算法。用户可以采用 DM 7 的加密引擎功能，将自己特殊的或高强度的加密算法按照 DM 7 提供的加密引擎标准接口要求进行封装，封装后的加密算法可以在 DM 7 的存储加密中按常规的方法进行使用，大大提高了数据的安全性。

10．客体重用

DM 7 内置的客体重用机制使数据库管理系统能够清扫被重新分配的系统资源，以保证数据信息不会因为资源的动态分配而泄露给未授权的用户。

1.1.5　易用性

DM 7 提供了一系列基于 Java 技术的多平台风格统一的全图形化客户端工具，通过 DM 7，用户可以与数据库进行交互，即操作数据库对象和从数据库获取信息，包括系统管理工具 Manager、数据迁移工具 DTS、性能监视工具 Monitor 等，同时支持基于 Web 的管理工具，该工具可以进行本地和远程联机管理。DM 7 提供的管理工具功能强大，界面友好，操作方便，能满足用户各种数据管理的需求。

1.1.6　兼容性

为保障用户现有应用系统上的投资，降低系统迁移的难度，DM 7 提供了许多与其他数据库系统兼容的特性，尤其针对 Oracle，DM 7 提供了全方位的兼容，以降低用户学习和迁移成本。

1.2　DM 7 体系结构

DM 7 数据库体系结构如图 1-1 所示，由数据库实例和物理存储结构组成。其中，实

例包括内存结构与后台进程，物理结构包括存储在磁盘上的数据文件、控制文件、日志文件、归档文件等。

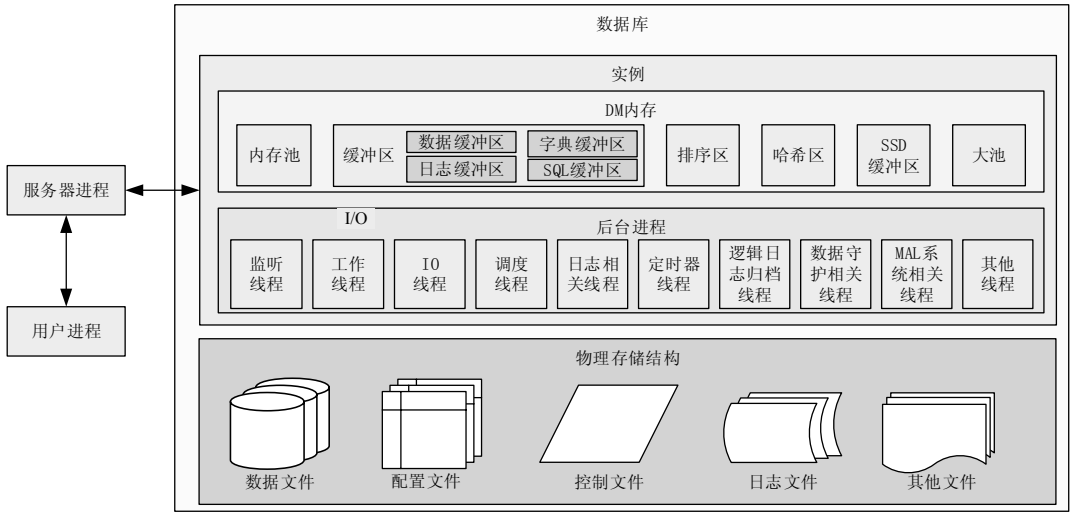

图 1-1　达梦数据库体系结构示意图

在达梦数据库中，数据的存储结构包括物理存储结构和逻辑存储结构两种。物理存储结构主要用于描述数据库外部数据的存储，即在操作系统中如何组织和管理数据，与具体的操作系统有关；逻辑存储结构主要描述数据库内部数据的组织和管理方式，与操作系统没有关系。物理存储结构是逻辑存储结构在物理上的、可见的、可操作的、具体的体现形式。

1.2.1　物理存储结构

物理存储结构描述了达梦数据库中的数据在操作系统中的组织和管理。典型的物理存储结构包括：用于进行功能设置的配置文件；用于记录文件分布的控制文件；用于保存用户实际数据的数据文件、重做日志文件、归档日志文件、备份文件；用来进行问题跟踪的跟踪日志文件等，如图 1-2 所示。

1．配置文件

配置文件是达梦数据库用来设置功能选项的一些文本文件的集合，配置文件以.ini 为扩展名，它们具有固定的格式，用户可以通过修改其中的某些参数取值实现特定功能项的启用和禁用，并针对当前系统运行环境设置更优的参数值以提升系统性能。

2．控制文件

每个达梦数据库都有一个名为 dm.ctl 的控制文件。控制文件是一个二进制文件，它记录了数据库必要的初始信息，其中主要包含以下内容。

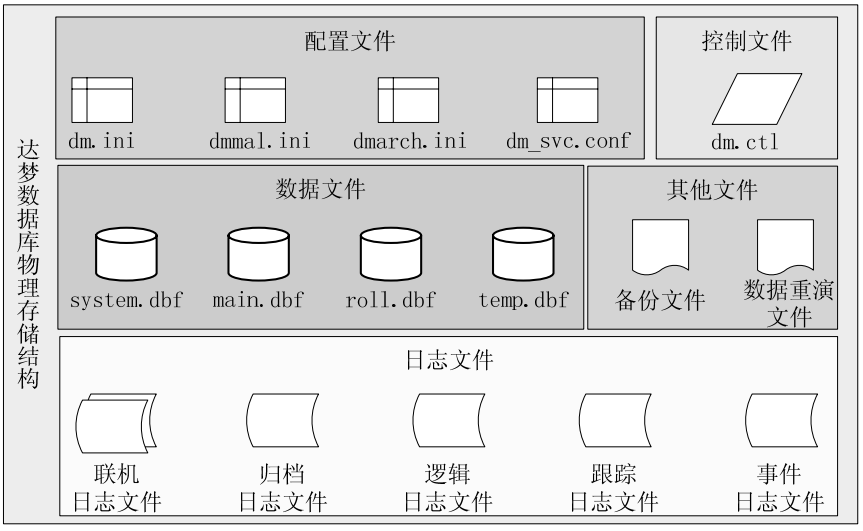

图 1-2　达梦数据库物理存储结构示意图

（1）数据库名称。

（2）数据库服务器模式。

（3）OGUID 唯一标识。

（4）数据库服务器版本。

（5）数据文件版本。

（6）表空间信息，包括表空间名，表空间物理文件路径等，记录了所有数据库中使用的表空间，以数组的方式保存起来。

（7）控制文件校验码，校验码由数据库服务器在每次修改控制文件后计算生成，保证控制文件的合法性，防止文件损坏及手工修改。

3．数据文件

数据文件以.dbf 为扩展名，它是数据库中最重要的文件类型，一个 DM 数据文件对应磁盘上的一个物理文件，数据文件是真实数据存储的地方，每个数据库至少有一个与之相关的数据文件。在实际应用中，通常有多个数据文件。

当 DM 的数据文件空间用完时，它可以自动扩展。可以在创建数据文件时通过 MAXSIZE 参数限制其扩展量，当然，也可以不限制。但是，数据文件的大小最终会受物理磁盘大小的限制。在实际使用中，一般不建议使用单个巨大的数据文件，为一个表空间创建多个较小的数据文件是更好的选择。

数据文件中还有两类特殊的文件：ROLL 和 TEMP 文件。

ROLL 文件用于保存系统的回滚记录，提供事务回滚时的信息。回滚文件是一个整段。每个事务的回滚页在回滚段中各自挂链，页内则顺序存放回滚记录。

TEMP 文件是临时数据文件，主要用于存放临时结果集，用户创建的临时表也存储在临时文件中。

4．重做日志文件

重做日志文件以.log 为扩展名。无论何时，在达梦数据库中添加、删除、修改对象，或者改变数据，都会将重做日志写入到当前的重做日志文件中。每个达梦数据库实例必须至少有两个重做日志文件，默认两个日志文件为 DAMENG01.log、DAMENG02.log，这两个文件循环使用。

理想情况下，数据库系统不会用到重做日志文件中的信息。然而现实世界总是充满了各种意外的，如电源故障、系统故障、介质故障，或者数据库实例进程被强制终止等，数据库缓冲区中的数据页会来不及写入数据文件。这样，在重启 DM 实例时，通过重做日志文件中的信息，就可以将数据库的状态恢复到发生意外时的状态。重做日志文件对于数据库是至关重要的。它们用于存储数据库的事务日志，以便系统在出现系统故障和介质故障时能够进行故障恢复。在达梦数据库运行过程中，任何修改数据库的操作都会产生重做日志，例如，当一条元组插入到一个表中的时候，插入的结果写入了重做日志，当删除一条元组时，删除该元组的事实也被写了进去，这样，当系统出现故障时，通过分析日志可以知道在故障发生前系统做了哪些动作，并可以重做这些动作使系统恢复到故障之前的状态。

日志文件分为联机日志文件和归档日志文件。以上所说的重做日志文件都指联机日志文件。在归档模式下，重做联机日志会被连续复制到归档日志中，这就生成了归档日志文件。联机日志文件指的是系统当前正在使用的日志文件，创建数据库时，联机日志文件通常被扩展至一定长度，其内容则被初始化为空，当系统运行时，该文件逐渐被产生的日志所填充。对日志文件的写入是顺序连续的。然而系统磁盘空间总是有限的，系统必须能够循环利用日志文件的空间，为了做到这一点，当所有日志文件空间被占满时，系统需要清空一部分日志以便重用日志文件的空间，为了保证被清空的日志所“保护”的数据在磁盘上是安全的，这里需要引入一个关键的数据库概念——检查点。当产生检查点时，系统将系统缓冲区中的日志和脏数据页都写入磁盘，以保证当前日志所“保护”的数据页都已安全写入磁盘，这样日志文件即可被安全重用。

5．归档日志文件

归档日志文件以归档时间命名，扩展名也是.log。达梦数据库可以在归档模式和非归档模式下运行。但只有在归档模式下运行时，达梦数据库在重做联机日志文件时才能生成归档日志文件。

采用归档模式会对系统的性能产生影响，然而系统在归档模式下运行会更安全，当出现故障时其丢失数据的可能性更小，这是因为一旦出现介质故障，如磁盘损坏，利用归档日志，系统可被恢复至故障发生的前一刻，也可以还原到指定的时间点，而如果没有归档日志文件，则只能利用备份进行恢复。

归档日志还是数据守护功能的核心，数据守护中的备机就是通过重做日志来完成与主机的数据同步的。

6．逻辑日志文件

如果在达梦数据库上配置了复制功能，复制源就会产生逻辑日志文件。逻辑日志文件是一个流式的文件，它有自己的格式，且不在页、簇和段的管理之下。

逻辑日志文件内部存储按照复制记录的格式，一条记录紧接着一条记录，存储着复制源端的各种逻辑操作，用于发送给复制目的端。

7．备份文件

备份文件以.bak 为扩展名，当系统正常运行时，备份文件不会起任何作用，它也不是数据库必须有的联机文件类型之一。然而，从来没有哪个数据库系统能够保证永远正确无误地运行，当数据库不幸出现故障时，备份文件就显得尤为重要了。

当客户利用管理工具或直接发出备份的 SQL 命令时，DM Server 会自动进行备份，并产生一个或多个备份文件，备份文件自身包含了备份的名称、对应的数据库、备份类型和备份时间等信息。同时，系统还会自动记录备份信息及该备份文件所处的位置，但这种记录是松散的，用户可根据需要将其复制至任何地方，并不会影响系统的运行。

8．跟踪日志文件

用户在 dm.ini 中配置 SVR_LOG 和 SVR_LOG_SWITCH_COUNT 参数后就会打开跟踪日志。跟踪日志文件是一个纯文本文件，以“dm_commit_日期_时间”命名，生成在 DM 安装目录的 log 子目录下，其内容包含系统各会话执行的 SQL 语句、参数信息、错误信息等。跟踪日志主要用于分析错误和分析性能问题，基于跟踪日志可以对系统运行状态有一个分析，例如，可以挑出系统现在执行速度较慢的 SQL 语句，进而对其进行优化。

打开跟踪日志会对系统的性能有较大影响，一般在查错和调优的时候才会打开，默认情况下系统是关闭跟踪日志的。

9．事件日志文件

达梦数据库系统在运行过程中，会在 log 子目录下产生一个“dm_实例名_日期”命名的事件日志文件。事件日志文件对达梦数据库运行时的关键事件进行记录，如系统启动、关闭、内存申请失败、I/O 错误等一些致命错误。事件日志文件主要用于系统出现严重错误时进行查看并定位问题。事件日志文件随着达梦数据库服务的运行一直存在。

10．数据重演文件

调用系统存储过程 SP_START_CAPTURE 和 SP_STOP_CAPTURE，可以获得数据重演文件。重演文件用于数据重演，存储了从抓取开始到抓取结束时，达梦数据库与客户端的通信消息。使用数据重演文件，可以多次重复抓取这段时间内的数据库操作，为系统调试和性能调优提供了另一种分析手段。

1.2.2 逻辑存储结构

达梦数据库逻辑存储结构描述了数据库内部数据的组织和管理方式。达梦数据库为数据库中的所有对象分配逻辑空间，并存放在数据文件中。在达梦数据库内部，所有的数据文件组合在一起被划分到一个或者多个表空间中，所有的数据库内部对象都存放在这些表空间中。同时，表空间被进一步划分为段、簇和页（也称块）。通过这种细分，可以使达梦数据库更加高效地控制磁盘空间的利用率。图 1-3 显示了这些数据结构之间的关系。

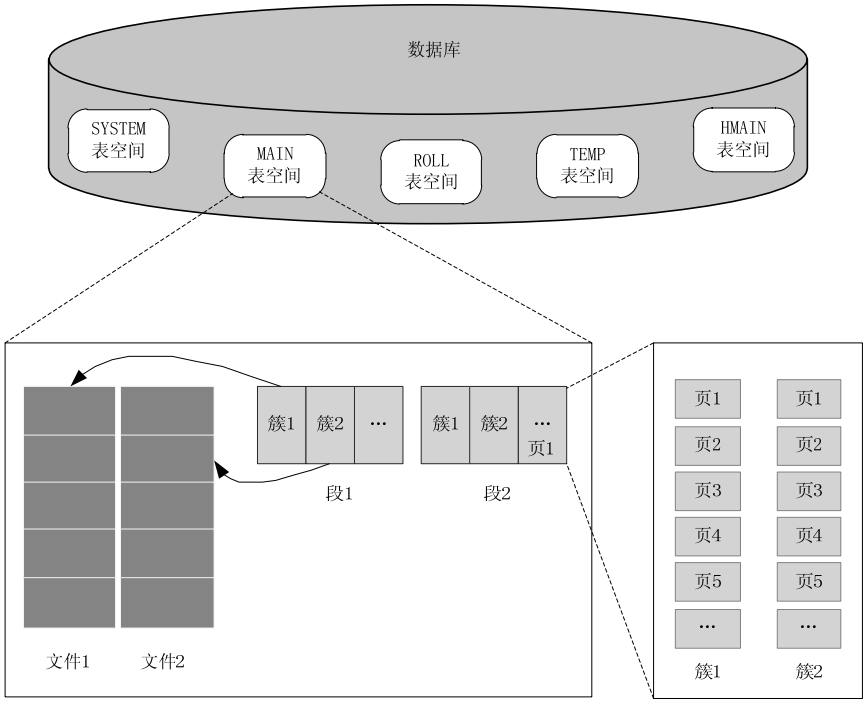

图 1-3　逻辑存储结构关系示意图

可以看出，在 DM 7 中存储的层次结构如下。

（1）系统由一个或多个表空间组成。

（2）每个表空间由一个或多个数据文件组成。

（3）每个数据文件由一个或多个簇组成。

（4）段是簇的上级逻辑单元，一个段可以跨多个数据文件。

（5）簇由磁盘上连续的页组成，一个簇总是在一个数据文件中。

（6）页是数据库中最小的分配单元，也是数据库中使用的最小的 I/O 单元。

1．表空间

在达梦数据库中，表空间由一个或者多个数据文件组成。达梦数据库中的所有对象在逻辑上都存放在表空间中，而物理上都存储在所属表空间的数据文件中。

在创建达梦数据库时，会自动创建 5 个表空间：SYSTEM 表空间、ROLL 表空间、MAIN 表空间、TEMP 表空间和 HMAIN 表空间。

（1）SYSTEM 表空间存放了有关达梦数据库的字典信息。

（2）ROLL 表空间完全由达梦数据库自动维护，用户无需干预。该表空间用来存放事务运行过程中执行 DML 操作之前的值，从而为访问该表的其他用户提供表数据的读一致性视图。

（3）MAIN 表空间在初始化库的时候，就会自动创建一个大小为 128MB 的数据文件 MAIN.dbf。在创建用户时，如果没有指定默认表空间，则系统自动指定 MAIN 表空间为用户默认的表空间。

（4）TEMP 表空间完全由达梦数据库自动维护。当用户的 SQL 语句需要磁盘空间来完成某个操作时，达梦数据库会从 TEMP 表空间中分配临时段，如创建索引、无法在内存中完成的排序操作、SQL 语句中间结果集以及用户创建的临时表等都会使用到 TEMP 表空间。

（5）HMAIN 表空间属于 HTS 表空间，完全由达梦数据库自动维护，用户无需干涉。当用户在创建 HFS 表时，在未指定 HTS 表空间的情况下，HMAIN 充当默认 HTS 表空间。

每一个用户都有一个默认的表空间。对于 SYSSSO、SYSAUDITOR 系统用户，默认的用户表空间是 SYSTEM，SYSDBA 的默认表空间为 MAIN，新创建的用户如果没有指定默认表空间，则系统自动指定 MAIN 表空间为用户默认的表空间。用户在创建表的时候，当指定了存储表空间 A，并且和当前用户的默认表空间 B 不一致时，表存储在用户指定的表空间 A 中，并且默认情况下，在这张表上建立的索引也将存储在 A 中，但是用户的默认表空间是不变的，仍为 B。一般情况下，建议用户自己创建一个表空间来存放业务数据，或者将数据存放在默认的用户表空间 MAIN 中，而不是将数据存放在 SYSTEM 表空间中。

2．页

数据页（也称数据块）是达梦数据库中最小的数据存储单元。页的大小对应物理存储空间上特定数量的存储字节，在达梦数据库中，页大小可以为 4KB、8KB、16KB 或者 32KB，用户在创建数据库时可以指定，默认大小为 8KB，一旦创建好了数据库，则在该库的整个生命周期内，页大小都不能够改变。图 1-4 显示了达梦数据库页的典型格式。

图 1-4　DM 数据页的典型格式

页头控制信息包含了页类型、页地址等信息。页的中部存放了数据，为了更好地利用数据页，在数据页的尾部专门留出一部分空间用于存放行偏移数组，行偏移数组用于标识页上的空间占用情况以便管理数据页自身的空间。

在绝大多数情况下，用户无需干预达梦数据库对数据页的管理。但是达梦数据库还是提供了选项供用户选择，使得在某些情况下可以为用户提供更佳的数据处理性能。

FILLFACTOR 是达梦数据库提供的一个与性能有关的数据页级存储参数，它指定一个数据页初始化后插入数据时最大可以使用空间的百分比（100），该值在创建表/索引时可以指定。

设置 FILLFACTOR 参数的值，是为了指定数据页中的可用空间百分比（FILLFACTOR）和可扩展空间百分比（100-FILLFACTOR）。可用空间用来执行更多的 INSERT 操作，可扩展空间用来为数据页保留一定的空间，以防止在今后的更新操作中增加列或者修改变长列的长度时，引起数据页的频繁分裂。当插入的数据占据的数据页空间百分比低于 FILLFACTOR 时，允许数据插入该页，否则将当前数据页中的数据分为两部分，一部分保留在当前数据页中，另一部分存入一个新页中。

对于 DBA 来说，使用 FILLFACTOR 时应该在空间和性能之间进行权衡。为了充分利用空间，用户可以设置一个很高的 FILLFACTOR 值，如 100，但是这可能会导致在后续更新数据时，频繁引起页分裂，而导致需要大量的 I/O 操作。为了提高更新数据的性能，可以设置一个相对较低（但不是过低）的 FILLFACTOR 值，使得后续执行更新操作时，可以尽量避免数据页的分裂，提升 I/O 性能，但这是以牺牲空间利用率来换取性能的提高。

3．簇

簇是数据页的上级逻辑单元，由同一个数据文件中 16 个或 32 个连续的数据页组成。在达梦数据库中，簇的大小由用户在创建数据库时指定，默认大小为 16。假定某个数据文件大小为 32MB，页大小为 8KB，则共有 32MB/8KB/16=256 个簇，每个簇的大小为 8KB×16=128KB。和数据页的大小一样，一旦创建好数据库，此后该数据库的簇的大小就不能够改变了。

1）分配数据簇

当创建一个表/索引的时候，DM 为表/索引的数据段分配至少一个簇，同时数据库会自动生成对应数量的空闲数据页，供后续操作使用。如果初始分配的簇中所有数据页都已经用完，或者新插入/更新数据需要更多的空间，达梦数据库将自动分配新的簇。在默认情况下，达梦数据库在创建表/索引时，初始分配 1 个簇，当初始分配的空间用完时，达梦数据库会自动扩展。

当达梦数据库的表空间为新的簇分配空闲空间时，首先在表空间中按文件从小到大的顺序在各个数据文件中查找可用的空闲簇，找到后进行分配；如果各数据文件都没有空闲簇，则在各数据文件中查找空闲空间足够的，将需要的空间先进行格式化，然后进行分配；如果各文件的空闲空间也不够，则选择一个数据文件进行扩充。

2）释放数据簇

对于用户数据表空间，在用户将一个数据段对应的表/索引对象 DROP 之前，该表对

应的数据段会保留至少 1 个簇不被回收到表空间中。在删除表/索引对象中的记录的时候，达梦数据库通过修改数据文件中的位图来释放簇，释放后的簇被视为空闲簇，可以供其他对象使用。当用户删除了表中所有记录时，达梦数据库仍然会为该表保留 1 或 2 个簇供后续使用。若用户使用 DROP 语句来删除表/索引对象，则此表/索引对应的段以及段中包含的簇全部收回，并供存储于此表空间的其他模式对象使用。

对于临时表空间，达梦数据库会自动释放在执行 SQL 过程中产生的临时段，并将属于此临时段的簇空间还给临时表空间。需要注意的是，临时表空间文件在磁盘所占大小并不会因此而缩减，用户可以通过系统函数 SF_RESET_TEMP_TS 来进行磁盘空间的清理。

对于回滚表空间，达梦数据库将定期检查回滚段，并确定是否需要从回滚段中释放一个或多个簇。

4．段

段是簇的上级逻辑分区单元，它由一组簇组成。在同一个表空间中，段可以包含来自不同文件的簇，即一个段可以跨越不同的文件。而一个簇以及该簇所包含的数据页则只能来自一个文件，是连续的 16 个或者 32 个数据页。由于簇的数量是按需分配的，因此数据段中的不同簇在磁盘上不一定连续。

1）数据段

段可以被定义成特定对象的数据结构，如表数据段或索引数据段。表中的数据以表数据段结构存储，索引中的数据以索引数据段结构存储。DM 以簇为单位给每个数据段分配空间，当数据段的簇空间用完时，达梦数据库就给该段重新分配簇，段的分配和释放完全由达梦数据库自动完成，可以在创建表/索引时设置存储参数来决定数据段的簇如何分配。

当用户使用 CREATE 语句创建表/索引时，DM 会创建相应的数据段。表/索引的存储参数用来决定对应数据段的簇如何被分配，这些参数将会影响与对象相关的数据段的存储与访问效率。对于分区表，每个分区使用单独的数据段来容纳所有数据，对于分区表上的非分区索引，使用一个索引数据段来容纳所有数据，而对于分区索引，每个分区使用一个单独索引数据段来容纳其数据。表的数据段和与其相关的索引段不一定要存储在同一表空间中，用户可以在创建表和索引时，指定不同的表空间存储参数。

2）临时段

在达梦数据库中，所有的临时段都创建在临时表空间中，这样可以分流磁盘设备的 I/O，也可以减少由于在 SYSTEM 或其他表空间内频繁创建临时数据段而造成的碎片。

当处理一个查询时，经常需要为 SQL 语句的解析与执行的中间结果准备临时空间。达梦数据库会自动地分配临时段的磁盘空间。例如，DM 在进行排序操作时就可能需要使用临时段，当排序操作可以在内存中执行，或设法利用索引就可以执行时，就不必创建临时段。对于临时表及其索引，达梦数据库也会为它们分配临时段。临时段的分配和释放完全由系统自动控制，用户不能手工进行干预。

3）回滚段

达梦数据库在回滚表空间的回滚段中保存了用于恢复数据库操作的信息。对于未提交事务，当执行回滚语句时，回滚记录被用来做回滚变更。在数据库恢复阶段，回滚记录被

用来做任何未提交变更的回滚。在多个并发事务运行期间，回滚段还为用户提供读一致性，所有正在读取受影响行的用户将不会看到行中的任何变动，直到他们事务提交后发出新的查询。达梦数据库提供了全自动回滚管理机制来管理回滚信息和回滚空间，自动回滚管理消除了管理回滚段的复杂性。此外，系统将尽可能保存回滚信息，来满足用户查询回滚信息的需要。事务被提交后，回滚数据不能再回滚或者恢复，但是从数据读一致性的角度出发，长时间运行查询可能需要这些早期的回滚信息来生成早期的数据页镜像，基于此，数据库需要尽可能长时间的保存回滚信息。达梦数据库会收集回滚信息的使用情况，并根据统计结果对回滚信息保存周期进行调整，数据库将回滚信息保存周期设为比系统中活动的最长的查询时间稍长。

1.2.3 实例

实例一般是由一个正在运行的 DM 后台进程（包含多个线程）以及一个大型的共享内存组成的。简单来说，实例就是操作达梦数据库的一种手段，是用来访问数据库的内存结构以及后台进程的集合。

达梦数据库存储在服务器的磁盘上，而 DM 实例存储于服务器的内存中。通过运行 DM 实例，可以操作达梦数据库中的内容。在任何时候，一个实例只能与一个数据库进行关联（装载、打开或者挂起数据库）。在大多数情况下，一个数据库也只有一个实例对其进行操作。但是在即将提供的 DM 共享磁盘高性能集群中，多个实例可以同时装载并打开一个数据库（位于一组由多台服务器共享的物理磁盘上）。此时，可以同时从多台不同的计算机访问这个数据库。

1．DM 后台进程

DM 服务器使用“对称服务器构架”的单进程、多线程结构。这种对称服务器构架在有效地利用了系统资源的同时又提供了较高的可伸缩性能，这里所指的线程即为操作系统的线程。服务器在运行时由各种内存数据结构和一系列的线程组成，线程分为多种类型，不同类型的线程完成不同的任务。线程通过一定的同步机制对数据结构进行并发访问和处理，以完成客户提交的各种任务。达梦数据库服务器是共享的服务器，允许多个用户连接到同一个服务器上，服务器进程称为共享服务器进程。DM 进程中主要包括监听线程、IO 线程、工作线程、调度线程、日志线程等。

1）监听线程

监听线程主要的任务是在服务器端口上进行循环监听，一旦有来自客户的连接请求，监听线程被唤醒并生成一个会话申请任务，加入工作线程的任务队列，等待工作线程进行处理。它在系统启动完成后才启动，并且在系统关闭时首先被关闭。为了保证在处理大量客户连接时系统具有较短的响应时间，监听线程比普通线程优先级更高。

2）工作线程

工作线程是 DM 服务器的核心线程，它从任务队列中取出任务，并根据任务的类型进行相应的处理，负责所有实际数据的相关操作。DM 7 的初始工作线程个数由配置文件指

定，随着会话连接的增加，工作线程也会同步增加，以保持每个会话都有专门的工作线程处理请求。为了保证用户所有请求及时响应，一个会话上的任务全部由同一个工作线程完成，这样减少了线程切换的代价，提高了系统效率。当会话连接超过预设的阈值时，工作线程数目不再增加，转而由会话轮询线程接收所有用户请求，加入任务队列，等待工作线程一旦空闲，从任务队列依次摘取请求任务处理。

3）IO 线程

在数据库活动中，IO 操作历来都是最为耗时的操作之一。当事务需要的数据页不在缓冲区中时，如果在工作线程中直接对那些数据页进行读写，将会使系统性能变得非常糟糕，而把 IO 操作从工作线程中分离出来是明智的做法。IO 线程的职责就是处理这些 IO 操作。通常情况下，DM Server 需要进行 IO 操作的时机主要有以下三种。

（1）需要处理的数据页不在缓冲区中，此时需要将相关数据页读入缓冲区。

（2）缓冲区满或系统关闭时，需要将部分脏数据页写入磁盘。

（3）检查点到来时，需要将所有脏数据页写入磁盘。

IO 线程在启动后，通常处于睡眠状态，当系统需要进行 IO 时，只需要发出一个 IO 请求，此时 IO 线程被唤醒以处理该请求，在完成该 IO 操作后继续进入睡眠状态。

IO 线程的个数是可配置的，可以通过设置 dm.ini 文件中的 IO_THR_GROUPS 参数来设置，默认情况下，IO 线程的个数是两个。同时，IO 线程处理 IO 的策略根据操作系统平台的不同会有很大差别，一般情况下，IO 线程使用异步的 IO 将数据页写入磁盘，此时，系统将所有的 IO 请求直接递交给操作系统，操作系统在完成这些请求后才通知 IO 线程，这种异步 IO 的方式使得 IO 线程需要直接处理的任务很简单，即完成 IO 后的一些收尾处理并发出 IO 完成通知，如果操作系统不支持异步 IO，此时 IO 线程需要完成实际的 IO 操作。

4）调度线程

调度线程用于接管系统中所有需要定时调度的任务。调度线程每秒轮询一次，负责的任务有以下一些。

（1）检查系统级的时间触发器，如果满足触发条件，则生成任务加到工作线程的任务队列中并由工作线程执行。

（2）清理 SQL 缓存、计划缓存中失效的项，或者超出缓存限制后淘汰不常用的缓存项。

（3）检查数据重演捕获持续时间是否到期，到期则自动停止捕获。

（4）执行动态缓冲区检查。根据需要动态扩展或动态收缩系统缓冲池。

（5）自动执行检查点。为了保证日志的及时刷盘，减少系统故障时恢复时间，根据 INI 参数设置的自动检查点执行间隔定期执行检查点操作。

（6）会话超时检测。当客户连接设置了连接超时时，定期检测是否超时，如果超时则自动断开连接。

（7）必要时执行数据更新页刷盘。

（8）唤醒等待的工作线程。

5）日志 FLUSH 线程

任何数据库的修改，都会产生重做（REDO）日志，为了保证数据故障恢复的一致性，REDO 日志的刷盘必须在数据页刷盘之前进行。事务运行时，会把生成的 REDO 日志保留

在日志缓冲区中，当事务提交或者执行检查点时，会通知 FLUSH 线程进行日志刷盘。由于日志具备顺序写入的特点，比数据页分散 IO 写入效率更高，因此，日志 FLUSH 线程和 IO 线程分开，能获得更快的响应速度，保证整体的性能。DM 7 的日志 FLUSH 线程进行了优化，在刷盘之前，对不同缓冲区内的日志进行合并，减少了 IO 次数，进一步提高了性能。如果系统配置了实时归档，在 FLUSH 线程日志刷盘前，会直接将日志通过网络发送到实时备机。如果配置了本地归档或者远程同步归档，则生成归档任务，通过日志归档线程完成。

6）日志归档线程

日志归档线程包含同步归档线程和异步归档线程，前者负责本地归档和远程同步归档任务，后者负责远程异步归档任务。如果配置了非实时归档，由日志 FLUSH 线程产生的任务会分别加入日志归档线程，日志归档线程负责从任务队列中取出任务，按照归档类型做相应归档处理。将日志 FLUSH 线程和日志归档线程分开的目的是减少不必要的效率损失，除了远程实时归档外，本地归档、远程同步归档、远程异步归档都可以脱离 FLUSH 线程来做，如果放在 FLUSH 线程中一起做，则会严重影响系统性能。

7）日志重做线程

为了提高故障恢复效率，DM 在故障恢复时采用了并行机制重做日志，日志重做线程就用于日志的并行恢复。通过 INI 参数 LOG_REDO_THREAD _NUM 可配置重做线程数，默认是两个线程。

8）日志 APPLY 线程

在配置了数据守护的系统中，创建了一个日志 APPLY 线程。当服务器作为备机时，每次接收到主机的物理 REDO 日志生成一个 APPLY 任务加入到任务队列，APPLY 线程从任务队列中取出一个任务在备机上将日志重做，并生成自己的日志，保持和主机数据的同步或一致，作为主机的一个镜像。备机数据对用户只读，可承担报表、查询等任务，均衡主机的负载。

9）定时器线程

在数据库的各种活动中，用户常常需要数据库在某个时间点开始进行某种操作，如备份；或者在某个时间段内反复进行某种操作等。定时器线程就是为这种需求而设计的。通常情况下，DM Server 需要进行定时操作的事件主要有以下几种。

（1）逻辑日志异步归档。

（2）异步归档日志发送（只在 PRIMARY 模式下，且 OPEN 状态下发送）。

（3）作业调度。

定时器线程启动之后，每秒检测一次定时器链表，查看当前的定时器是否满足触发条件，如果满足，则把执行权交给设置好的任务，如逻辑日志异步归档等。默认情况下，达梦服务器启动的时候，定时器线程是不启动的。用户可以通过设置 dm.ini 中的 TIMER_INI 参数为 1 来设置定时器线程在系统启动时启动。

10）逻辑日志归档线程

逻辑日志归档用于 DM 7 的数据复制，目的是加快异地访问的响应速度，包含本地逻辑日志归档线程和远程逻辑日志归档线程。当配置了数据复制后，系统才会创建这两个线程。

（1）本地逻辑日志归档线程：本地归档线程从本地归档任务列表中取出一个归档任务，生成到逻辑日志，并将逻辑日志写入到逻辑日志文件中。如果当前逻辑日志的远程归档类型是同步异地归档并且当前的刷盘机制是强制刷盘，那么就生成一个异地归档任务加入到临时列表中。

（2）远程逻辑日志归档线程：远程归档线程从远程归档任务列表中取出一个归档任务，并根据任务的类型进行相应的处理。任务的类型包括同步发送和异步发送。

11）数据守护相关线程

在配置了数据守护的观察器上，会创建观察器的实时检测线程、同步检测线程，实现主备机之间的故障检测、故障切换以及恢复。在配置了守护进程的数据守护方案中，数据库实例还会创建 UDP 消息的广播和接收线程，负责实例和守护进程之间的通信，实现数据守护功能。

12）MAL 系统相关线程

MAL 系统是 DM 内部高速通信系统，基于 TCP/IP 协议实现。服务器的很多重要功能都是通过 MAL 系统实现通信的，如数据守护、数据复制、MPP、远程日志归档等。MAL 系统内部包含一系列线程，有 MAL 监听线程、MAL 发送工作线程、MAL 接收工作线程等。

13）其他线程

事实上，达梦数据库系统中不止以上这些线程，在一些特定的功能中会有不同的线程，如回滚段清理 PURGE 线程、审计写文件线程、重演捕获写文件线程等，这里不再一一列出。

2．DM 内存

数据库管理系统是一种对内存申请和释放操作频率很高的软件，如果每次对内存的使用都使用操作系统函数来申请和释放，效率会比较低，加入自己的内存管理是 DBMS 所必需的。通常内存管理系统会带来以下好处。

（1）申请、释放内存效率更高。

（2）能够有效地了解内存的使用情况。

（3）易于发现内存泄露和内存写越界的问题。

达梦数据库管理系统的内存结构主要包括内存池、缓冲区、排序区、哈希区等。根据系统中子模块的不同功能，对内存进行了上述划分，并采用了不同的管理模式。

1）内存池

DM Server 的内存池指的是共享内存池，根据内存使用情况的不同，对共享内存池的使用采用了两种工作方式：HEAP 和 VPOOL。共享内存池用于解决 DM Server 对于小片内存的申请与释放问题。系统在运行过程中，经常会申请与释放小片内存，而向操作系统申请和释放内存时需要发出系统调用，此时可能会引起线程切换，降低系统运行效率。采用共享内存池可一次向操作系统申请一片较大内存，即为内存池，当系统在运行过程中需要申请内存时，在共享内存池内进行申请，当用完该内存后再释放，即归还给共享内存池。当系统采用较好的策略管理共享内存池时，小片内存的申请与释放不会对系统影响太大。

这种方式还有一个优点，可以比较容易地检测系统是否存在内存泄漏。DM 系统管理员可以通过 DM Server 的配置文件（dm.ini）来对共享内存池的大小进行设置，共享池的参数为 MEMORY_POOL，该配置默认为 40MB。而 HEAP 与 VPOOL 使用的是共享内存池中的内存，所以一般情况下，HEAP 与 VPOOL 两种方式对内存申请的大小不会超过 MEMORY_POOL 值。

（1）HEAP：HEAP 工作方式采用了堆的思想。每次申请内存时，都是从堆顶上申请的，如果不够，则继续向共享内存池中申请内存页，然后加入到 HEAP 中，继续供系统的申请使用，这样 HEAP 的长度可以无限增长下去；释放 HEAP 时，可以释放堆顶上的内存页，也可以释放整个 HEAP。使用内存堆来管理小片内存的申请有一个特点，每次申请小片内存以后，不能单独对这片内存进行释放，也就是不用关心这片内存何时释放，它在堆释放时统一释放，能有效防止内存泄露的发生。

（2）VPOOL：VPOOL 的工作方式主要采用了“伙伴系统”的思想进行管理。申请的 VPOOL 内存分为私有和公有两种。私有 VPOOL 只提供给某个单独功能模块使用，公有 VPOOL 则提供给那些需要共享同一资源而申请的模块，所以对公有 VPOOL 需要进行保护，而私有 VPOOL 则不需要。VPOOL 和 HEAP 的区别在于，VPOOL 申请出去的每片内存都可以单独地进行释放。

2）缓冲区

（1）数据缓冲区：数据缓冲区是 DM Server 在将数据页写入磁盘之前以及从磁盘上读取数据页之后，数据页所存储的地方。这是 DM Server 至关重要的内存区域之一，将其设定得太小，会导致缓冲页命中率低，磁盘 IO 频繁；将其设定得太大，又会导致操作系统内存本身不够用。系统启动时，首先根据配置的数据缓冲区大小向操作系统申请一片连续内存并将其按数据页大小进行格式化，并置入“自由”链中。数据缓冲区存在三条链来管理被缓冲的数据页；一条是“自由”链，用于存放目前尚未使用的内存数据页；一条是“LRU”链，用于存放已被使用的内存数据页（包括未修改和已修改的）；还有一条即为“脏”链，用于存放已被修改过的内存数据页。LRU 链对系统当前使用的页按其最近是否被使用的顺序进行了排序。这样当数据缓冲区中的自由链被用完时，从 LRU 链中淘汰部分最近未使用的数据页，能够较大程度地保证被淘汰的数据页在最近不会被用到，减少 IO。在系统运行过程中，通常存在一部分“非常热”（反复被访问）的数据页，将它们一直留在缓冲区中，对系统性能会有好处。对于这部分数据页，数据缓冲区开辟了一个特定的区域用于存放它们，以保证这些页不参与一般的淘汰机制，可以一直留在数据缓冲区中。

① 类别：DM Server 中有四种类型的数据缓冲区，分别是 NORMAL、KEEP、FAST 和 RECYCLE。其中，用户可以在创建表空间或修改表空间时，指定表空间属于 NORMAL 或 KEEP 缓冲区。RECYCLE 缓冲区供临时表空间使用，FAST 缓冲区根据用户指定的 FAST_POOL_PAGES 和 FAST_ROLL_PAGES 大小由系统自动进行管理，用户不能指定使用 RECYCLE 和 FAST 缓冲区的表或表空间。NORMAL 缓冲区主要是提供给系统处理的一些数据页，没有特定指定缓冲区的情况下，默认缓冲区为 NORMAL；KEEP 的特性是对缓冲区中的数据页很少或几乎不怎么淘汰，主要针对用户的应用是否需要经常处在内存当中，如果是，则可以指定缓冲区为 KEEP。DM Server 提供了可以更改这些缓冲区大小的

参数，用户可以根据自己应用需求情况，指定 dm.ini 文件中 BUFFER(80MB)、KEEP(8MB)、RECYCLE(64MB)、FAST_POOL_PAGES(0)和 FAST_ROLL_ PAGES(0)值（括号中为默认值），这些值分别对应 NORMAL 缓冲区大小、KEEP 缓冲区大小、RECYCLE 缓冲区大小、FAST 缓冲区页面数和 FAST 缓冲区回滚页面数。

② 读多页：在需要进行大量 I/O 的应用当中，DM 之前版本的策略是每次只读取一页。如果知道用户需要读取表的大量数据，当读取到第一页时，可以猜测用户可能需要读取这页的下一页，在这种情况下，一次性读取多页就可以减少 I/O 次数，从而提高了数据的查询、修改效率。DM Server 提供了可以读取多页的参数，用户可以指定这些参数来调整数据库运行效率的最佳状态。在 DM 配置文件 dm.ini 中，可以指定参数 MULTI_PAGE_GET_NUM 大小（默认值为 16 页），以控制每次读取的页数。

如果用户没有设置较适合的参数 MULTI_PAGE_GET_NUM 值大小，有时可能会给用户带来更差的效果。如果 MULTI_PAGE_GET_NUM 太大，每次读取的页可能大多不是以后所用到的数据页，这样不仅会增加 I/O 的读取，而且每次都会做一些无用的 I/O，所以系统管理员需要衡量好自己的应用需求，给出最佳方案。

（2）日志缓冲区：日志缓冲区是用于存放重做日志的内存缓冲区。为了避免由于直接的磁盘 IO 而使系统性能受到影响，系统在运行过程中产生的日志并不会立即被写入磁盘，而是和数据页一样，先将其放置到日志缓冲区中。那么为何不在数据缓冲区中缓存重做日志而要单独设立日志缓冲区呢？其主要基于以下原因。

重做日志的格式同数据页完全不一样，无法进行统一管理；

重做日志具备连续写的特点；

在逻辑上，写重做日志比数据页 IO 优先级更高。

DM Server 提供了参数 LOG_BUF_SIZE 对日志缓冲区大小进行控制，日志缓冲区所占用的内存是从共享内存池中申请的，单位为页数量，且大小必须为 2 的 N 次方，否则采用系统默认大小 256 页。

（3）字典缓冲区：字典缓冲区主要存储一些数据字典信息，如模式信息、表信息、列信息、触发器信息等。每次对数据库的操作都会涉及数据字典信息，访问数据字典信息的效率直接影响到相应的操作效率，如进行查询语句，就需要相应的表信息、列信息等，这些字典信息如果都在缓冲区里，则直接从缓冲区中获取即可，否则需要 I/O 才能读取到这些信息。DM 7 采用的是将部分数据字典信息加载到缓冲区中，并采用 LRU 算法进行字典信息的控制。缓冲区大小，如果设置的太大，会浪费宝贵的内存空间；如果太小，可能会频繁地进行淘汰，该缓冲区配置参数为 DICT_BUF_SIZE，默认的配置大小为 5MB。DM 7 采用缓冲部分字典对象，这会影响效率吗？数据字典信息访问存在热点现象，并不是所有的字典信息都会被频繁的访问，所以按需加载字典信息并不会影响到实际的运行效率。但是如果在实际应用中涉及对分区数较多的水平分区表访问，如上千个分区，那么就需要适当调大 DICT_BUF_SIZE 参数值。

（4）SQL 缓冲区：SQL 缓冲区提供在执行 SQL 语句过程中所需要的内存，包括计划、SQL 语句和结果集缓存。很多应用当中都存在反复执行相同 SQL 语句的情况，此时可以使用缓冲区保存这些语句和它们的执行计划，这就是计划重用。这样带来的好处是加快了

SQL 语句执行效率，但同时给内存增加了压力。DM Server 在配置文件 dm.ini 中提供了参数来支持是否需要计划重用，参数为 USE_PLN_POOL，当指定为 1 时，启动计划重用；否则禁止计划重用。DM 还提供了参数 CACHE_POOL_SIZE（单位为 MB）来改变 SQL 缓冲区大小，系统管理员可以设置该值以满足应用需求，默认值为 10MB。

3）排序区

排序区提供数据排序所需要的内存缓冲空间。当用户执行 SQL 语句时，常常需要进行排序，所使用的内存就是排序缓冲区提供的。在每次排序过程中，都先申请内存，排序结束后再释放内存。DM Server 提供了参数来指定排序缓冲区的大小，参数 SORT_BUF_SIZE 在 DM 配置文件 dm.ini 中，系统管理员可以设置其大小以满足需求，由于该值是由系统内部排序算法和排序数据结构决定的，建议使用默认值 2MB。

4）哈希区

DM 7 提供了为哈希连接而设定的缓冲区，不过该缓冲区是虚拟缓冲区。之所以说是虚拟缓冲，是因为系统没有真正创建特定属于哈希缓冲区的内存，而在进行哈希连接时，对排序的数据量进行了计算。如果计算出的数据量大小超过了哈希缓冲区的大小，则使用 DM 7 创新外存哈希方式；如果没有超过哈希缓冲区的大小，实际上使用的还是 VPOOL 内存池进行的哈希操作。DM Server 在 dm.ini 中提供了参数 HJ_BUF_SIZE 来进行控制，由于该值的大小可能会限制哈希连接的效率，所以建议保持默认值，或设置为更大的值。

除了提供 HJ_BUF_SIZE 参数外，DM Server 还提供了创建哈希表个数的初始化参数，其中，HAGR_HASH_SIZE 表示处理聚集函数时创建哈希表的个数，建议保持默认值 100000。

5）SSD 缓冲区

固态硬盘采用闪存作为存储介质，因没有机械磁头的寻道时间，在读写效率上比机械磁盘具有优势。在内存、SSD 磁盘、机械磁盘之间，符合存储分级的条件。为提高系统执行效率，DM Server 将 SSD 文件作为内存缓存与普通磁盘之间的缓冲层，称为“SSD 缓存”。DM Server 在 dm.ini 中提供参数 SSD_BUF_SIZE 和 SSD_FILE_PATH 来配置 SSD 缓冲，SSD_BUF_SIZE 指定缓冲区的大小，单位是 MB，DM Server 根据该参数创建相应大小的文件作为缓冲区使用；SSD_FILE_PATH 指定该文件所在的文件夹路径，管理员需要保证设置的路径是位于固态磁盘上的。

默认 SSD 缓冲区是关闭的，即 SSD_BUF_SIZE 为 0。若要配置 SSD 缓冲区，将其设置为大于 0 的数并指定 SSD_FILE_PATH 即可，根据存储分级的概念，建议将 SSD_BUF_SIZE 配置为 BUFFER_SIZE 的 2 倍左右。

1.2.4 工作机制

实例一般是由一组正在运行的 DM 后台进程/线程以及一个大型的共享内存组成的。简单来说，实例就是操作达梦数据库的一种手段，是用来访问数据库的内存结构以及后台进程的集合。

达梦数据库存储在服务器的磁盘上，而 DM 实例存储于服务器的内存中。通过运行

DM 实例，可以操作达梦数据库中的内容。在任何时候，一个实例只能与一个数据库进行关联（装载、打开或者挂起数据库）。在大多数情况下，一个数据库也只有一个实例对其进行操作。但是在即将提供的 DM 共享磁盘高性能集群中，多个实例可以同时装载并打开一个数据库（位于一组由多台服务器共享的物理磁盘上）。此时，可以同时从多台不同的计算机访问这个数据库。

1.3　DM 7 常用工具

DM 7 提供了功能丰富的系列工具，包括控制台工具、管理工具、性能监视工具、数据迁移工具、数据库配置助手、PL/SQL 调试工具和审计分析工具等。

1. 控制台工具

控制台工具是管理和维护数据库的基本工具。通过使用控制台工具，数据库管理员可以完成以下功能：服务器参数配置、管理 DM 服务、脱机备份与还原、查看系统信息、查看许可证信息、数据守护配置与状态监视。其界面如图 1-5 所示。

2. 管理工具

DM 管理工具是达梦系统最主要的图形界面工具，用户通过它可以与数据库进行交互——操作数据库对象和从数据库获取信息。其主要功能包括：服务器管理、数据库实例管理、模式对象管理、表对象管理、外部表对象管理、索引对象管理、视图对象管理、物化视图对象管理、存储过程对象管理、函数对象管理、序列对象管理、触发器对象管理、包对象管理、类对象管理、同义词对象管理、全文索引对象管理、外部链接对象管理、角色权限管理、用户权限管理、安全信息管理、表空间对象管理、备份恢复管理、工具包管理、数据复制管理、数据守护管理、作业调度管理等。其界面如图 1-6 所示。

3. 性能监视工具

性能监视工具是达梦系统管理员用来监视服务器的活动和性能情况，并对系统参数进行调整的客户端工具，它允许系统管理员在本机或远程监视服务器的运行状态。其界面如图 1-7 所示。

4. 数据迁移工具

数据迁移工具提供了主流大型数据库迁移到 DM、DM 迁移到主流大型数据库、DM 到 DM、文件迁移到 DM 以及 DM 迁移到文件的功能。DM 数据迁移工具采用向导方式引导用户通过简单的步骤完成需要的操作。其界面如图 1-8 所示。

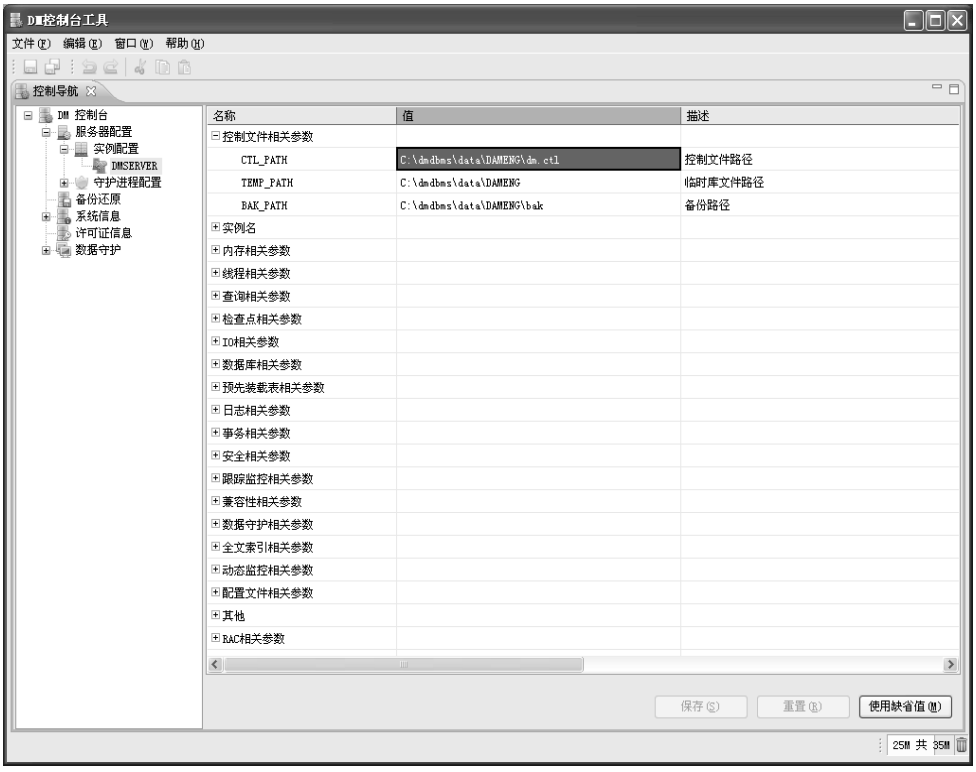

图 1-5　控制台工具界面

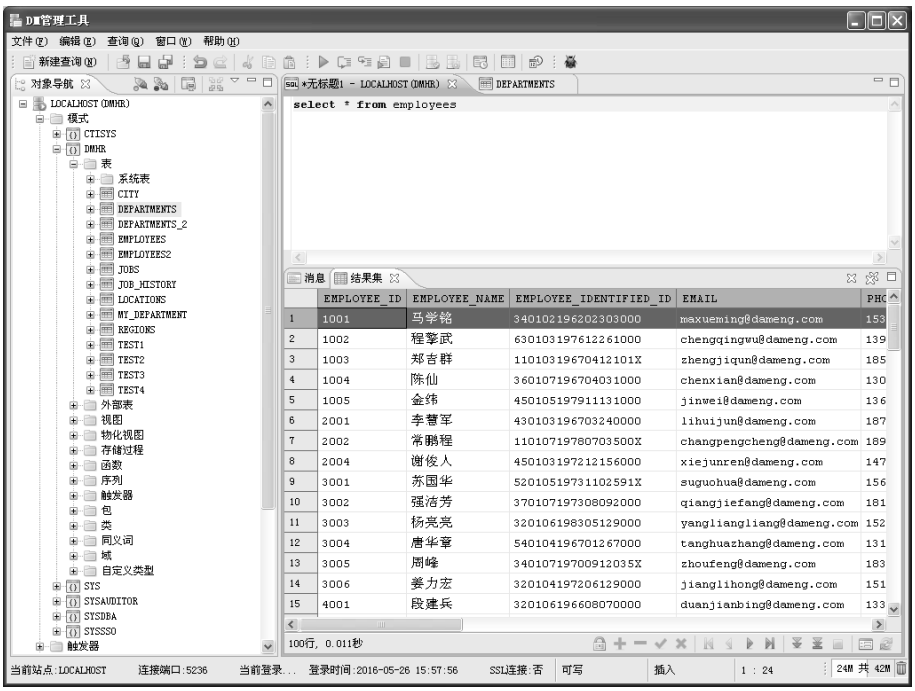

图 1-6　管理工具界面

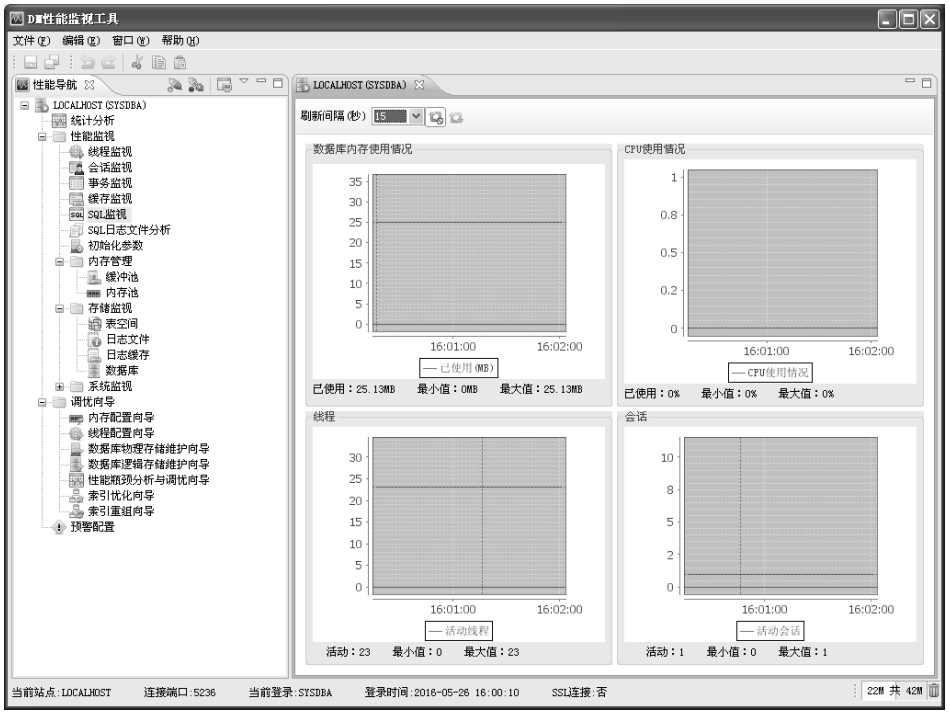

图 1-7　性能监视工具界面

图 1-8　数据迁移工具界面

5. 数据库配置助手

数据库配置助手是达梦数据库提供的数据库配置工具，以便用户在创建数据库的时候，能够通过图形界面设置初始化数据库的参数。其界面如图 1-9 所示。

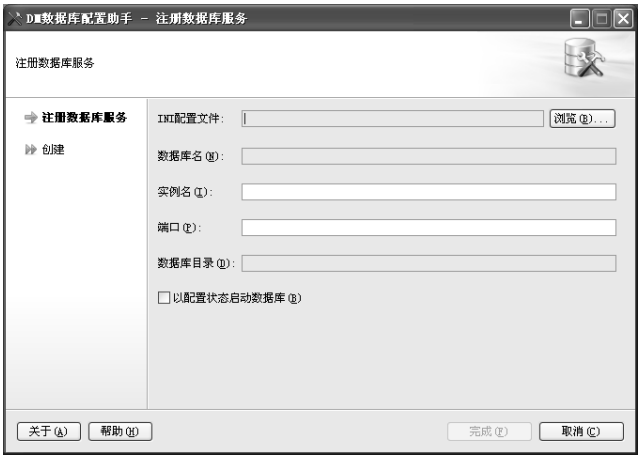

图 1-9　数据库配置助手界面

6. 审计分析工具

DM 审计分析工具是数据库审计日志查看的基本工具。通过使用审计分析工具，数据库审计员可以完成审计规则的创建与修改，以及审计记录的查看与导出等功能。DM 审计分析工具的界面如图 1-10 所示。

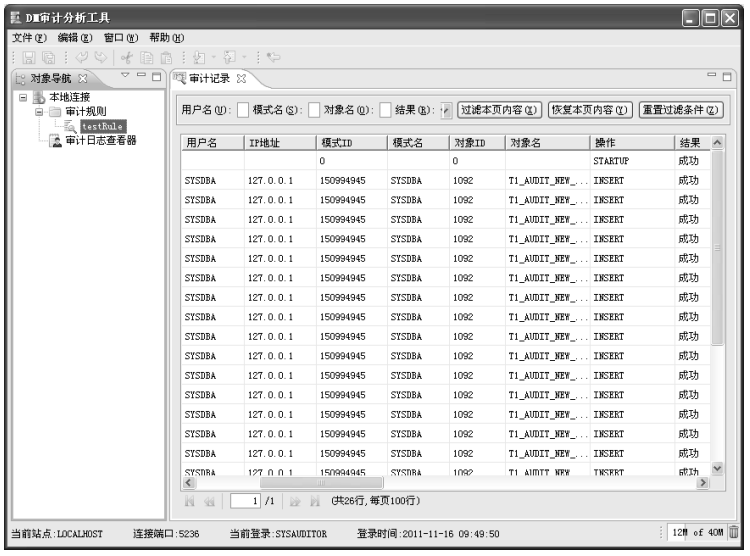

图 1-10　审计分析工具界面

2

第 2 章

安装与卸载

达梦数据库管理系统是基于客户机/服务器方式的数据库管理系统，可以安装在多种计算机操作系统平台上，典型的操作系统有 Windows、Linux、Solaris 和 AIX 等。

DM 在代码级全面支持 32 位和 64 位系统，DM 既可运行在 32 位操作系统上，又可运行在 64 位操作系统上，尤其在 64 位系统上能充分利用 64 位系统资源（如能充分利用更大容量的内存）使性能更优。同时，由于 DM 客户端程序主要使用 Java 编写，具有良好的跨平台特性，可运行在上述操作系统上。客户端程序所用的操作系统与服务器所用的操作系统无关。

对于不同的操作系统平台，达梦数据库安装与卸载存在一定的差异性，本章主要介绍 Windows 平台和 Linux 平台下达梦数据库的安装和卸载。

2.1 Windows 下 DM 7 安装与卸载

达梦数据库几乎支持所有版本的 Windows 操作系统，如 Windows 2000、Windows 2003、Windows XP、Windows Vista、Windows 7、Windows 8 和 Windows 10 等。在 Windows 操作系统上安装达梦数据库较为方便，只需检查软硬件环境是否满足达梦数据安装的基本要求，即可运行达梦数据库安装程序，通过向导式的图形安装界面完成安装。

2.1.1 安装前准备

1．硬件环境

安装达梦数据库前应检查硬件配置是否满足基本要求。安装达梦数据库所需的硬件基

本配置见表 2-1。

表 2-1　安装 DM 所需的硬件基本配置

硬　件	基 本 配 置
CPU	Intel Pentium 4（建议 Pentium 4 1.6GB 以上）处理器
内存	256MB（建议 512MB 以上）
硬盘	5GB 以上可用空间
网卡	10MB 以上支持 TCP/IP 协议的网卡
光驱	32 倍速以上光驱
显卡	支持 1024×768×256 以上彩色显示
显示器	SVGA 显示器
键盘/鼠标	普通键盘/鼠标

由于 DM 是基于客户机/服务器方式的大型数据库管理系统，一般基于网络环境部署，达梦数据库软件和客户端软件分别部署在数据库服务器和客户端计算机上，所以硬件环境通常包括网络环境（如一个局域网）。当然，也可部署于单机上，即达梦数据库软件和客户端软件部署在一台计算机上。

2．软件环境

达梦数据库支持几乎所有 Windows 版本的操作系统，但需注意以下几点。

（1）系统盘可用空间建议大于 1GB。

（2）关闭正在运行的防火墙、杀毒软件等。

（3）在安装 32 位版本数据库之前，还应保证系统时间在 1970 年 1 月 1 日 00:00:00 到 2038 年 1 月 19 日 03:14:07 之间。

（4）若系统中已安装 DM，重新安装前，应在备份数据后，完全卸载原来的系统。

2.1.2　服务器端软件安装

在 Windows 系统下安装达梦数据库，应使用 Administrator 账户或其他拥有管理员权限的账户进行安装。所以在运行达梦数据库安装程序前，应使用 Administrator 或者其他拥有管理员权限的账户登录。

在 Windows 系列操作系统下 DM 的安装是一样的，在 Windows 环境下安装 DM 服务器端软件和安装一般软件十分相似，下面以 Windows XP 为例描述整个安装过程，其他的 Windows 环境可以参考此安装过程。用户可根据安装向导完成 DM 服务器端软件的安装，DM 服务器端软件的具体安装步骤如下。

步骤 1：用户在确认 Windows 系统已正确安装和网络系统能正常运行的情况下，将 DM 安装光盘放入光驱中，会自动进入安装界面，如已将安装程序复制至本地硬盘中，则应运行“setup.exe”文件。程序将检测当前计算机系统是否已经安装其他版本达梦数据库系统，如果存在其他版本达梦数据库，将弹出提示对话框，如图 2-1 所示。建议卸载其他

版本达梦数据库后再安装，如果继续安装，将弹出语言与时区选择对话框，如图 2-2 所示，请根据系统配置选择相应语言与时区，默认为“简体中文”与“GTM+08:00 中国标准时间”，单击“确定”按钮继续安装。

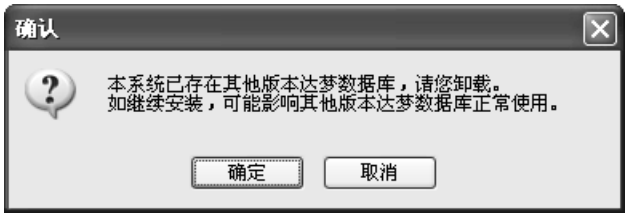

图 2-1　版本检测提示信息

图 2-2　语言和时区选择界面

之后会进入安装欢迎界面，如图 2-3 所示，单击“开始”按钮继续安装。

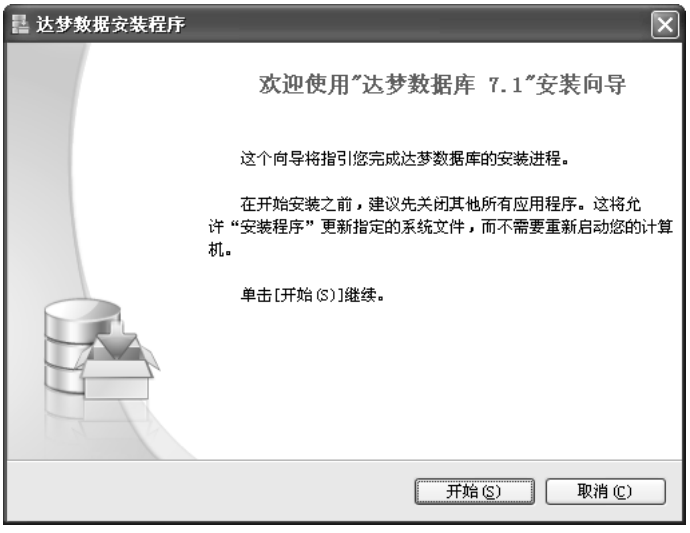

图 2-3　安装欢迎界面

步骤 2：接受许可证协议，如图 2-4 所示。在安装和使用 DM 之前，该安装程序需要用户阅读许可协议条款，用户如接受该协议，则选中“接受”单选按钮，并单击“下一步”按钮继续安装；用户若选中“不接受”单选按钮，将无法进行安装。

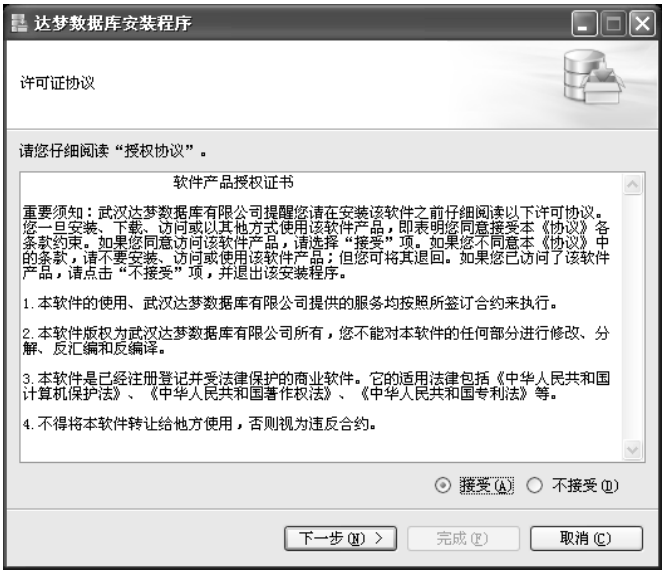

图 2-4　许可证协议界面

步骤 3：查看版本信息，用户可通过图 2-5 所示界面查看 DM 服务器、客户端等各组件相应的版本信息，读者获得的软件版本可能比此版本更新。单击“下一步”按钮继续安装。

图 2-5　版本信息界面

步骤 4：验证 Key 文件。图 2-6 所示为 Key 文件验证界面，用户单击“浏览”按钮，选取 Key 文件，安装程序将自动验证 Key 文件信息。如果是合法的 Key 文件且在有效期内，用户可以单击“下一步”按钮继续安装。

图 2-6　Key 文件验证界面

步骤 5：选择安装方式。如图 2-7 所示，DM 安装程序提供四种安装方式：“典型安装”、“服务器安装”、“客户端安装”和“自定义安装”，用户可根据实际情况灵活地选择。

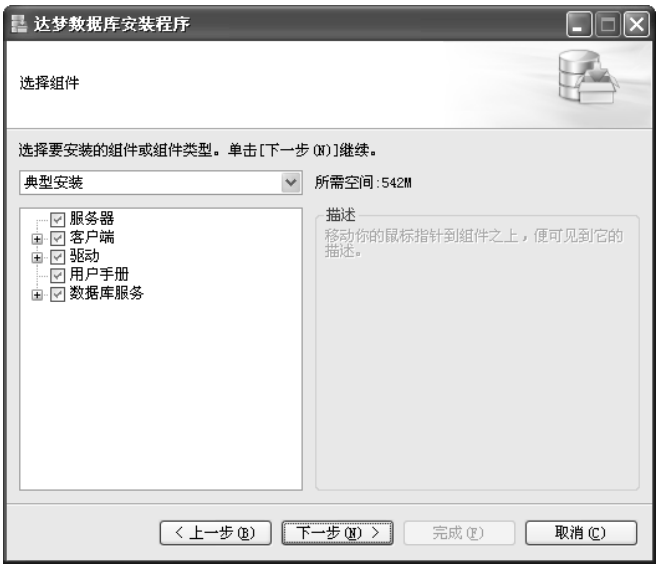

图 2-7　安装方式选择界面

用户若想安装服务器端、客户端所有组件，则选择“典型安装”，单击“下一步”按钮继续；用户若只想安装 DM 服务器端组件，则选择“服务器安装”，单击“下一步”按钮继续；用户若只想安装所有的客户端组件，则选择“客户端安装”，单击“下一步”按钮继续；用户若想自定义安装，则选择“自定义安装”，选中自己需要的组件，在本安装过程中，要安装服务器端组件，请确认选中服务器组件选项，并单击“下一步”按钮继续。

一般的，作为服务器端的计算机只需选择“服务器安装”选项，特殊情况下，服务器端的计算机也可以作为客户机使用，此时，计算机必须安装相应的客户端软件。

步骤 6：设置安装目录。如图 2-8 所示，设置达梦数据库安装目录。达梦数据库默认安装在 C:\dmdbms 目录下，用户可以通过单击“浏览”按钮自定义安装目录。

说明：安装路径里的目录名由英文字母、数字和下画线等组成，不支持包含空格的目录名，建议不要使用中文字符等。

步骤 7：设置达梦“开始菜单”文件夹。选择达梦快捷方式在“开始”菜单中的文件夹名称，默认为“达梦数据库”，如图 2-9 所示。

图 2-8　设置安装目录界面

图 2-9　设置“开始菜单”文件夹

步骤 8：安装前小结。图 2-10 显示用户即将进行的安装的有关信息，如产品名称、版本信息、安装类型、安装目录、是否初始化数据库、可用空间、可用内存等信息，用户检查无误后单击“安装”按钮，开始安装软件。

图 2-10　安装前小结界面

步骤 9：正式安装。图 2-11 所示为正在安装界面。

图 2-11　安装界面

步骤 10：初始化数据库。如用户在选择安装组件时选中了服务器组件，数据库自身安装过程结束时，将会提示是否初始化数据库，如图 2-12 所示。若用户未安装服务器组件，安装完成后，单击“完成”按钮将直接退出。

图 2-12　数据库安装完成界面

若用户选中“创建数据库实例”单选按钮，单击“开始”按钮将弹出数据库配置助手，如图 2-13 所示，初次安装数据库需要初始化数据库。

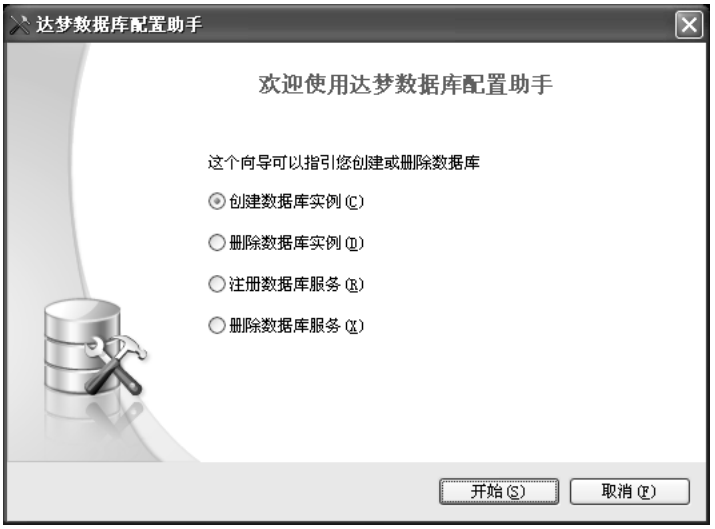

图 2-13　数据库配置助手界面

步骤 11：选择操作方式。在图 2-13 所示界面中用户可选择创建数据库实例、删除数据库实例、注册数据库服务和删除数据库服务等操作方式，但初次安装数据库应选中“创建数据库实例”单选按钮，单击“开始”按钮。

步骤 12：选择数据库模板。系统提供三套数据库模板供用户选择：一般用途、联机分析处理和联机事务处理，用户可根据自身的用途选择相应的模板，如图 2-14 所示。

图 2-14　选择模板界面

步骤 13：指定数据库所在目录。用户可通过浏览或输入的方式设置数据库所在目录，如图 2-15 所示。

图 2-15　设置数据库所在目录

步骤 14：设置数据库标识。用户可输入数据库名称、实例名、端口号等参数，如图 2-16 所示。

图 2-16　设置数据库标识

步骤 15：设置数据库文件所在位置。用户可通过选择或输入确定数据库控制文件、数据库日志文件的所在位置，并可通过右侧功能按钮，对文件进行添加或删除，如图 2-17 所示。

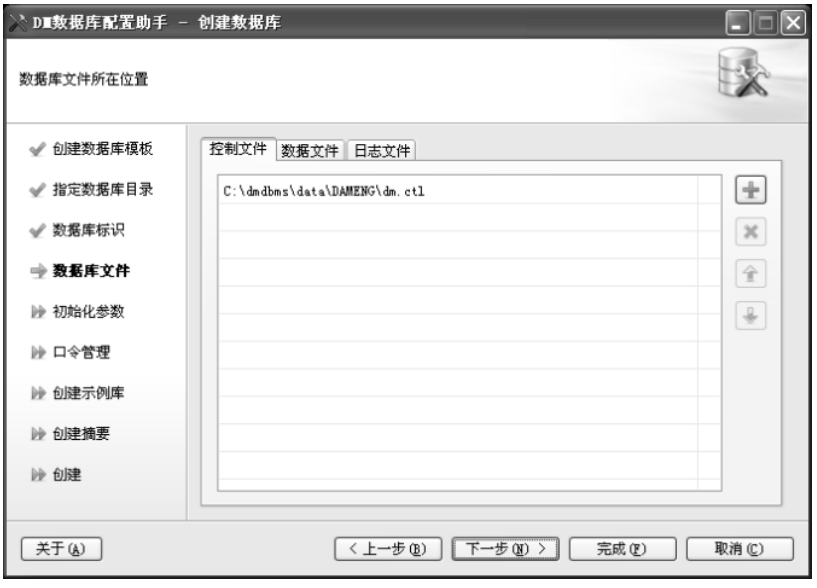

图 2-17　设置数据库文件所在位置

步骤 16：设置数据库初始化参数。用户可输入数据库相关参数，如簇大小、页大小、日志文件大小、是否大小写敏感、是否使用 Unicode 等，如图 2-18 所示。

图 2-18　设置数据初始化参数

步骤 17：设置数据库口令。用户可输入 SYSDBA、SYSAUDITOR 的密码，对默认口令进行更改，如果安装版本为安全版，则会增加 SYSSSO 用户的密码修改，如图 2-19 所示。

图 2-19　设置数据库口令

步骤 18：选择是否创建示例库。请读者选中创建示例库复选框，如图 2-20 所示。

图 2-20　选择是否创建示例库

步骤 19：查看创建数据库的摘要，在安装数据库之前，将显示用户通过数据库配置工具设置的相关参数，如图 2-21 所示。

图 2-21　查看创建数据库的摘要

步骤 20：创建数据库实例，如图 2-22 所示。

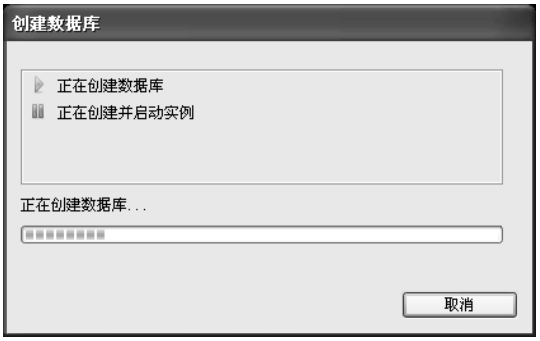

图 2-22　正创建数据库实例界面

步骤 21：创建数据库完成。数据库创建完成后，将进入如图 2-23 所示数据库完成界面，并可通过“DM 服务查看器”工具查看 DM 相关服务，如图 2-24 所示。

图 2-23　创建数据库完成界面

图 2-24　查看 DM 相关服务

2.1.3 客户端软件安装

DM 在 Windows 平台下提供的客户端程序主要有以下几种。

（1）管理工具：Manager。

（2）数据迁移工具：Dts。

（3）控制台工具：Console。

（4）性能监控工具：Monitor。

（5）审计分析工具：Analyzer。

（6）ODBC 3.0 驱动程序：dodbc.dll。

（7）JDBC 3.0 驱动程序：DM 7JdbcDriver.jar。

（8）OLEDB 2.7 驱动程序：doledb.dll。

（9）C Language Tools：一组 C 语言开发的命令行工具。

注意：命令行工具包括 disql、dminit、DM Server 等，以及预编译工具 PreCompiler（ProC 编译工具/环境）等。

DM 客户端软件的安装和 DM 服务器端的安装步骤基本一致，先把 DM 安装光盘放入光驱中，将自动执行 DM 安装光盘上的安装程序，或复制至硬盘中运行“setup.exe”文件。

步骤 1～4：参考 2.1.2 小节，客户端软件的安装与服务器端软件的安装步骤类似。

步骤 5：选择安装方式，如图 2-7 所示。用户若想安装所有的客户端组件，则选择“客户端安装”，单击“下一步”按钮继续；用户也可以选择“自定义安装 ”，根据需要选择要安装的客户端组件，单击“下一步”按钮继续。

说明：DM 的编程接口的动态库文件（dmdpi.dll）在安装过程中是自动安装的。

其余步骤参考 2.1.2 小节，客户端软件的安装与服务器端软件的安装步骤类似。

因为客户端安装不需要初始化数据库，所以在安装过程中不执行 2.1.2 小节所述的步骤 10 及之后的所有步骤。

安装完毕，安装程序自动在“开始”菜单下添加“达梦数据库”选项。用户可以单击相应的快捷方式启动已经安装的客户端软件。安装程序把客户端工具安装在目标路径的 tool 目录下，用户也可以直接找到目标路径，启动相应的客户端软件。

2.1.4 许可证安装

用户安装 DM 时，可导入相应的许可证（License）。License 的载体是一个加密文件 dm.key，内容是达梦公司对用户使用 DM 软件的授权。DM 软件安装后，如果需要得到更多授权的用户，可联系达梦公司获取相应的 License，并按照下面的操作方法进行安装。

用户获得 License 文件 dm.key 后，首先将达梦服务器关闭，然后将 dm.key 复制到 DM 安装目录下 DM 服务器所在的子目录中即可。

操作方法如下。

首先，打开 DM 服务器所在的目录。

例如，当 DM 安装目录为 C:\dmdbms 时，DM 服务器所在的目录为 C:\dmdbms\bin。

其次，将达梦服务器关闭，再将 dm.key 复制到该目录下，替换原来的 dm.key 文件即可。

说明：该目录下原有的 dm.key 文件请事先做好备份。

2.1.5　卸载

达梦数据库软件的卸载和普通软件卸载类似，只需通过向导式的操作界面即可完成达梦数据库软件的卸载，具体操作步骤如下。

步骤 1：备份数据。由于卸载达梦数据库服务端软件将致使数据库无法使用，因此在卸载达梦服务端软件时应先备份数据。备份数据方法请参考本书后续章节。

步骤 2：选择“开始-程序-达梦数据库-卸载”选项，开始卸载。

步骤 3：确认是否卸载达梦数据库。卸载之前将会弹出如图 2-25 所示的“确认”对话框，防止用户误操作，用户可单击“确定”按钮确认卸载。

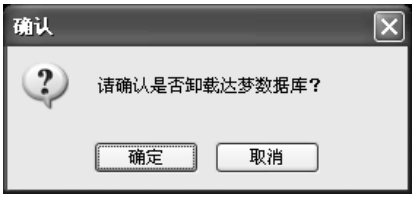

图 2-25　卸载确认界面

步骤 4：提示卸载信息。如图 2-26 所示，达梦数据库卸载会提示“删除系统上已安装过的功能部件，但不会删除安装后创建的文件夹和文件”，并提示安装目录，用户可直接单击“卸载”按钮开始卸载数据库系统，如存在数据库服务正在运行，会再次弹出如图 2-27 所示的确认卸载对话框，卸载过程如图 2-28 所示。

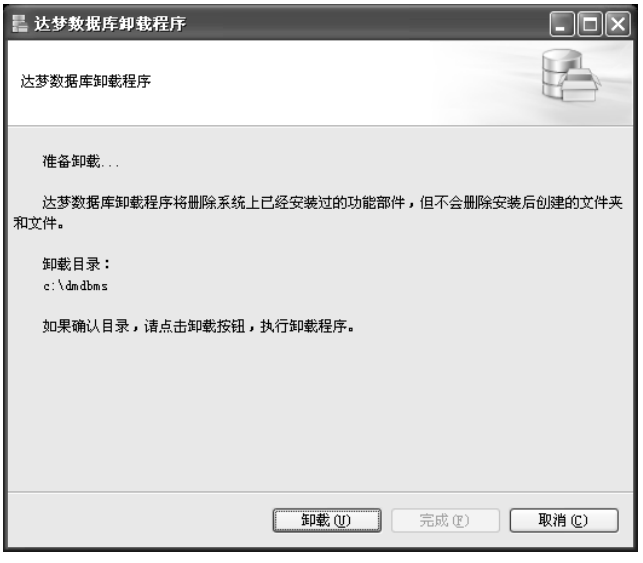

图 2-26　数据库卸载提示界面

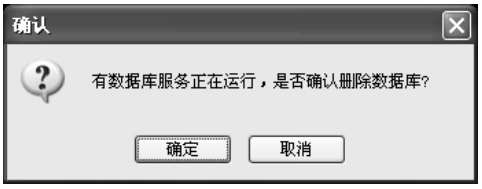

图 2-27　确认删除数据库对话框

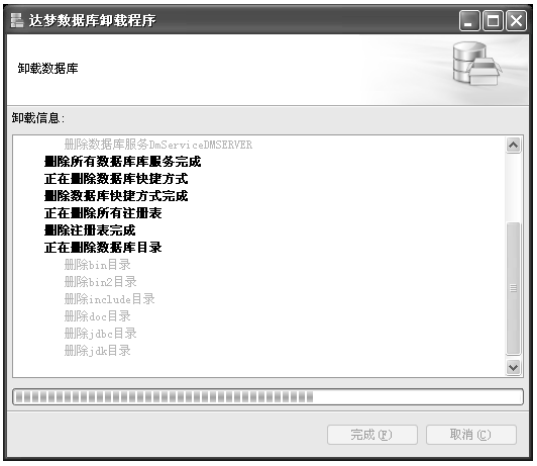

图 2-28　正在卸载数据库

步骤 5：达梦数据库卸载完成，删除数据库安装完成后创建的文件和文件夹。达梦数据库卸载完成后会进入如图 2-29 所示界面。同时，达梦数据库卸载并不删除数据库安装完成后创建的文件和文件夹，需手工删除，如需手工删除 C:\dmdmbs 目录及其下的所有文件。

图 2-29　达梦数据库卸载完成界面

2.2　Linux 下 DM 7 安装与卸载

Linux 下 DM 安装与卸载也基于图形化界面，其与 Windows 下 DM 的安装与卸载基本类似，不同之处主要体现在安装之前的准备工作上。

2.2.1　安装前准备

1．硬件环境

Linux 下达梦数据库的安装所需硬件环境与 Windows 下安装要求一致，请参考 2.1.1 小节所述硬件环境内容。

2．软件环境

Linux 下达梦数据库的安装所需软件环境与 Windows 下安装要求类似，可参考 2.1.1 小节所述软件环境内容，但 Linux 需内核 2.6 及以上，并已安装 KDE/GNOME 桌面环境，建议预先安装 UNIXODBC 组件。

3．其他准备工作

1）检查临时目录的大小

由于安装程序产生的临时文件默认使用/tmp 目录，为了使安装程序正常运行，用户应该保证/tmp 有大于 600MB 的剩余空间。用户可以使用以下命令进行查询：

```
df -h /tmp
```

如果/tmp 目录的剩余空间不足，用户可以扩充/tmp 目录的空间，也可以通过设置环境变量 DM_INSTALL_TMPDIR 指定安装程序的临时目录。如通过以下语句设置临时目录：

```
export   DM_INSTALL_TMPDIR=/opt/tmp
```

或修改配置文件（如/etc/profile）：

```
DM_INSTALL_TMPDIR=/opt/tmp
export DM_INSTALL_TMPDIR
```

2）创建安装用户

为了减少对操作系统的影响，用户不应该以 root 用户来安装和运行达梦数据库，用户可以在安装之前为达梦数据库创建一个专用的系统用户。对达梦数据库的初学者或对 Linux 不熟悉的用户，可直接使用 root 用户，因直接使用 root 用户安装更简单。

以下步骤为创建安装用户步骤，只作为参考提示，具体步骤及操作请以安装时的系统为准，具体细节可向系统管理员咨询。以 root 用户登录后，在终端依次执行以下命令。

（1）创建用户组：

```
groupadd dinstall
```

（2）创建安装用户：

```
useradd -g dinstall -m -d /home/dmdba -s /bin/bash dmdba
```

（3）初始化用户密码：

```
passwd dmdba
```

之后通过系统提示进行密码设置。

3）创建安装程序临时目录

安装时默认使用 /tmp 目录作为安装程序的临时目录，如果用户要通过环境变量

DM_INSTALL_TMPDIR 指定安装程序临时目录，需先创建此目录，并确保 dmdba 系统用户对此目录有写入权限（如使用 root 用户安装，则无需执行此步骤）。假设创建安装程序临时目录为/opt/tmp，则在终端中应执行以下命令：

```
mkdir -p /opt/tmp
chown -R dmdba:dinstall /opt/tmp
chmod -R 664 /opt/tmp
```

4）检查系统资源限制

在 Linux、Solaris、AIX 和 HP-UNIX 等系统中，因为 ulimit 命令的存在，会对程序使用操作系统资源进行限制。为了使达梦数据库正常运行，建议用户检查 ulimit 的参数。

首先使用 dmdba 系统用户进行登录（如使用 root 安装 DM，请使用 root 用户登录操作系统），运行 ulimit -a 命令查询所有参数，重点关注以下参数。

（1）data seg size (kbytes, -d)：建议用户设置为 1048576（1GB）以上或 unlimited（无限制），此参数过小将导致数据库启动失败。

（2）file size (blocks, -f)：建议用户设置为 unlimited（无限制），此参数过小将导致数据库安装或初始化失败。

（3）open files (-n)：建议用户设置为 65536 以上或 unlimited（无限制）。

（4）virtual memory (kbytes, -v)：建议用户设置为 1048576（1GB）以上或 unlimited（无限制），此参数过小将导致数据库启动失败。

通常情况下仅需修改 open files 参数，其余参数均符合要求。有多种修改 open files 参数的方法，如在/etc/profile 文件后追加“ulimit -n 65536”命令，重启操作系统即可完成该参数的修改。具体修改方法请以操作的计算机为准，或向 Linux 管理原咨询。

2.2.2 服务器端软件安装

DM 的安装向导在 Linux 平台下与 Windows 下非常相似。

安装过程如下。

步骤 1：加载光驱或复制安装文件。

以安装达梦数据库的用户（如 dmdba 或 root）登录到 Linux 系统，将 DM 安装光盘放入光驱，然后加载（mount）光驱。一般可以通过执行下面的命令来加载光驱：

```
mount   /dev/cdrom      /mnt/cdrom
```

这里假定光驱对应的文件为/dev/cdrom 且目标路径/mnt/cdrom 已存在，如目标路径不存在，可先执行 mkdir -p /mnt/cdrom 命令创建目录。

步骤 2：启动安装程序。

在 Linux 的 KDE 或 GNOME 图形界面（可通过 startx 命令进入）下，单击光盘下的 DMInstall.bin 启动安装程序（或直接进入/mnt/cdrom 目录后执行命令./DMInstall.bin），将进入 Linux 下的 DM 安装界面，如图 2-30 所示。

如果不能够运行，请检查当前用户对 DMInstall.bin 是否具有执行权限，如果没有则使用 chmod 命令增加当前用户的执行权限或直接向 Linux 系统管理员咨询。

图 2-30　Linux 下的 DM 安装界面

步骤 3：参照 2.1.2 小节 Windows 下 DM 服务器端软件的安装步骤进行安装和初始化数据库。

当以非 root 系统用户安装时，安装页面将提示配置环境变量。用户请手动配置环境变量，将安装目录的 bin 文件夹路径添加到 LD_LIBRARY _PATH 环境变量中，具体操作步骤请咨询 Linux 管理员。

图 2-31　Linux 下配置环境变量提示

同时，当以非 root 系统用户安装时，当安装进度完成时将会弹出对话框，提示使用 root 用户执行相关命令，用户可根据对话框的说明完成相关操作，之后可关闭此对话框，单击“完成”按钮结束安装，如图 2-32 所示。

步骤 4：卸载光驱。

安装完后一般要卸载光驱才能取出光盘。与前面的 mount 光驱命令相对应，可以使用下面的命令来卸载光驱：

```
umount  /mnt/cdrom
```

图 2-32　Linux 下执行配置脚本提示

2.2.3　客户端软件安装

DM 在 Linux 平台下提供的客户端程序主要有以下几种。

（1）管理工具：Manager。

（2）数据迁移工具：Dts。

（3）控制台工具：Console。

（4）性能监控工具：Monitor。

（5）审计分析工具：Analyzer。

（6）ODBC 3.0 驱动程序。

（7）JDBC 3.0 驱动程序。

（8）C Language Tools：一组 C 语言开发的命令行工具。

安装过程如下。

1．加载光驱

参见 2.2.2 小节 Linux 平台下 DM 服务器端软件的加载光驱过程。

2．启动安装程序

参见 2.2.2 小节 Linux 平台下 DM 服务器端软件的启动安装程序过程。

3．安装客户端程序

参照 2.1.3 小节 Windows 下 DM 客户端软件的安装步骤进行安装。

4．卸载光驱

参见 2.2.2 小节 Linux 平台下 DM 服务器端软件的卸载光驱过程。

说明：DM 客户端软件所用的操作系统与服务器端软件所用的操作系统无关。

Windows 下的客户端软件也可以访问 Linux 下的 DM 服务器。因此也可以将客户端软件安装在 Windows 下而不安装在 Linux 下。

2.2.4　命令行方式安装 DM 服务器和客户端软件

在现实中，许多主机尤其是一些运行 UNIX 操作系统的主机上没有图形化操作系统界面，为了使 DM 能够在这些主机上顺利运行，也可以直接在英文字符界面下，采用命令行的方式进行安装，其过程与图形化的安装步骤一致。安装前，建议首先建立达梦数据库安装用户，具体步骤请参见 Linux 平台下 DM 图形化界面安装步骤。参见 2.2.1 小节安装前准备工作。

在终端进入到安装程序所在文件夹时，执行./DMInstall.bin –i 并按提示操作，如图 2-33 所示。

图 2-33　字符界面安装 DM

2.2.5　许可证安装

用户获得 License 文件 dm.key 后，首先将达梦服务器关闭，然后将 dm.key 复制到 DM 安装目录下，DM 服务器所在的子目录中即可。这与 Windows 下 License 的安装类似。操

作方法如下。

首先，找到 DM 服务器所在的目录，方法是以 root 用户或安装用户登录到 Linux 系统，启动终端，执行以下命令即可进入 DM 服务器程序安装的目录（注意，假设安装目录为 /opt/dmdbms）：

```
cd  /opt/dmdbms/bin
```

其次，先将达梦服务器关闭，再将 dm.key 文件复制到该目录中，替换原有的 dm.key 即可。

说明：该目录下原有的 dm.key 文件请事先做好备份。

2.2.6 卸载

Linux 下达梦数据库软件的卸载与 Windows 下的卸载类似，用户可在 Linux 操作系统中执行卸载快捷方式，或者执行安装目录下的 uninstall.sh 脚本卸载。程序将会弹出提示对话框确认是否卸载程序，具体卸载过程参见 Windows 下的卸载过程。在 Linux 下，使用非 root 用户卸载完成时，将会弹出对话框，提示使用 root 用户执行相关命令，用户可根据对话框的说明完成相关操作，之后可关闭此对话框，单击“完成”按钮结束卸载，如图 2-34 所示。

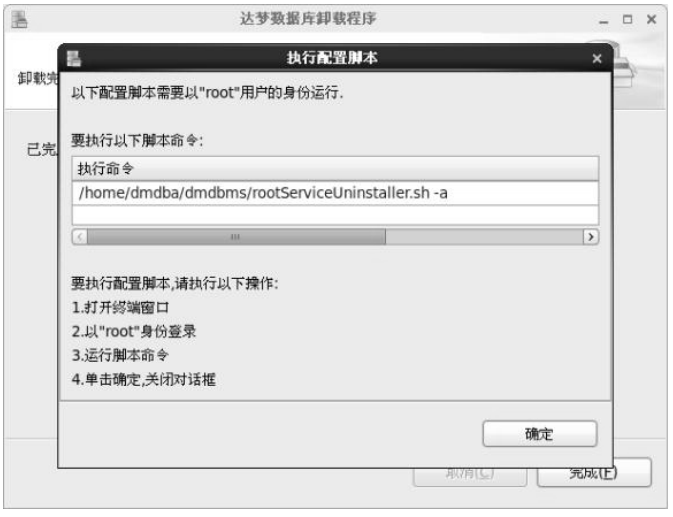

图 2-34　Linux 下的卸载提示

3

第 3 章

表空间管理

DM 表空间是对达梦数据库的逻辑划分，一个数据库有多个表空间，一个表空间对应着磁盘上一个或多个数据库文件。从物理存储结构上讲，数据库的对象，如表、视图、索引、序列、存储过程等存储在磁盘的数据文件中，从逻辑存储结构上讲，这些数据库对象都存储在表空间当中，因此表空间是创建其他数据库对象的基础。

根据表空间的用途不同，表空间又可以细分为基本表空间、临时表空间、大表空间等，本章重点介绍表空间的创建、修改、删除等日常管理操作。

从本章开始，很多例子之间有关联性，从前向后阅读本书有助于理解和重现这些例子，如果在重现例子过程中出现了问题，可以查看“附加说明”。

3.1 创建表空间

简单地说，创建表空间过程就是在磁盘上创建一个或多个数据文件的过程，这些数据文件被达梦数据库管理系统控制和使用，这些数据文件所占的磁盘存储空间归数据库所有，这些数据文件将用于存储表、视图、索引等内容，这些数据文件可以占据固定的磁盘空间，也可以随着存储数据量的增加而不断扩展。我们既可以运用 SQL 命令来创建表空间，也可以通过 DM 管理工具来创建表空间，下面分别进行介绍。

3.1.1 用 SQL 命令创建表空间

1. 语法格式

为了更好地掌握语法格式，达梦数据库采用分级描述的方法，先描述总体的语法格式，

然后描述子元素的语法格式。语法格式中：

< > 表示一个语法对象。

::= 表示定义符，用来定义一个语法对象。定义符左边为语法对象，右边为相应的语法描述。

| 表示语法选项在实际的语句中只能出现一个。

{ } 表示语法选项在实际的语句中可以出现 0～N 次（N 为大于 0 的自然数）。

[ ] 表示语法选项在实际的语句中可以出现 0 或 1 次。

在达梦数据库中，完整的 SQL 语法要素非常多，表述和阅读非常困难，在不影响读者学习理解这些 SQL 命令的前提下，本书对某些 SQL 命令做了一些简化，仅保留常用要素。希望了解完整语法的读者，可以阅读《DM 7_SQL 语言使用手册》电子版。

创建表空间的 SQL 命令格式如下：

```
CREATE TABLESPACE <表空间名> <数据文件子句>[<数据页缓冲池子句>][<存储加密子句>];
```

其中，各子句具体语法如下：

```
<数据文件子句> ::= DATAFILE <文件说明项>{,<文件说明项>}
<文件说明项> ::= <文件路径> [ MIRROR <文件路径>] SIZE <文件大小>[<自动扩展子句>]
<自动扩展子句> ::= AUTOEXTEND <ON [<每次扩展大小子句>][<最大大小子句> |OFF>
<每次扩展大小子句> ::= NEXT <扩展大小>
<最大大小子句> ::= MAXSIZE <文件最大大小>
<数据页缓冲池子句> ::= CACHE = <缓冲池名>
<存储加密子句> ::= ENCRYPT WITH <加密算法> BY <加密密码>
```

创建表空间时必须指定表空间的名称和表空间使用的数据文件，当一个表空间有多个数据文件时，在数据文件子句中依次列出。数据页缓冲池子句是可选项，默认值为“NORMAL”，存储加密子句是可选项，默认不加密。

语法格式中的各项参数的详细说明见表 3-1。

表 3-1 参数说明

参 数	说 明
<表空间名>	表空间名称最大长度为 128 字节
<文件路径>	指明新生成的数据文件在操作系统下的路径＋新数据文件名。数据文件的存放路径符合 DM 安装路径的规则，且该路径必须是已经存在的
MIRROR	数据文件镜像，用于在数据文件出现损坏时替代数据文件进行服务。<文件路径>必须是绝对路径，必须在建库时开启页校验的参数 page_check
<文件大小>	整数值，指明新增数据文件的大小(单位 MB)，取值为 4096×页大小～2147483647×页大小

2. 应用举例

【例 3-1】创建一个名为 EXAMPLE 的表空间，包含一个数据文件 EXAMPLE.DBF，初始大小为 128MB。

```
SQL>CREATE TABLESPACE example DATAFILE 'C:\dmdbms\data\DAMENG\EXAMPLE.DBF' SIZE 128;
```

在 SQL 语句书写过程中，标点符号均为英文标点符号，关键字通常大写，语法对象通常小写。文件大小的默认单位为 MB，在命令中只写数据文件大小的阿拉伯数字即可。

【例 3-2】创建一个名称为 TS1 的表空间，包含两个数据文件，其中，TS101.DBF 文件初始大小为 128MB，可自动扩展，每次扩展 4MB，最大扩展至 1024MB，TS102.DB 文件初始大小为 256MB，不能自动扩展。

（1）创建 TS1 表空间：

```
SQL>CREATE TABLESPACE ts1 DATAFILE 'C:\dmdbms\data\DAMENG\TS101.DBF' SIZE 128
AUTOEXTEND ON NEXT 4 MAXSIZE 1024,'C:\dmdbms\data\DAMENG\TS102.DBF' SIZE 256
AUTOEXTEND OFF;
```

（2）查询 TS1 表空间：

```
SQL>SELECT file_name, autoextensible FROM dba_data_files WHERE tablespace_name='TS1';
行号        FILE_NAME                              AUTOEXTENSIBLE
---------- ------------------------------ ------------------------------------------------------------
1          C:\dmdbms\data\DAMENG\TS101.DBF        YES
2          C:\dmdbms\data\DAMENG\TS102.DBF        NO
```

这个例子说明，一个逻辑意义上的表空间可以包含磁盘上的多个物理数据文件。

【例 3-3】创建一个名称为 TS2 的表空间，包含一个数据文件 TS201.DBF，初始大小为 512MB。

```
SQL>CREATE TABLESPACE ts2 DATAFILE 'c:\dmdbms\data\DAMENG\TS201.DBF' SIZE 512;
```

这个例子说明，创建表空间的命令中，除了一些必要的参数外，其他参数可以省略，采用默认值。

3. 附加说明

（1）创建表空间的用户必须具有 DBA 权限。

（2）表空间名在服务器中必须唯一。

（3）一个表空间最多可以拥有 256 个数据文件。

3.1.2　用管理工具创建表空间

【例 3-4】创建一个名为 EXAMPLE2 的表空间，包含一个数据文件 EXAMPLE2.DBF，初始大小为 128MB。

步骤 1：在图 3-1 中右击“表空间”节点，在弹出的快捷菜单中选择“新建表空间”选项，弹出“新建表空间”对话框，如图 3-2 所示。

步骤 2：在图 3-2 中，在“表空间名”文本框中输入表空间的名称 EXMPLE2。对话框中的参数说明见表 3-2。

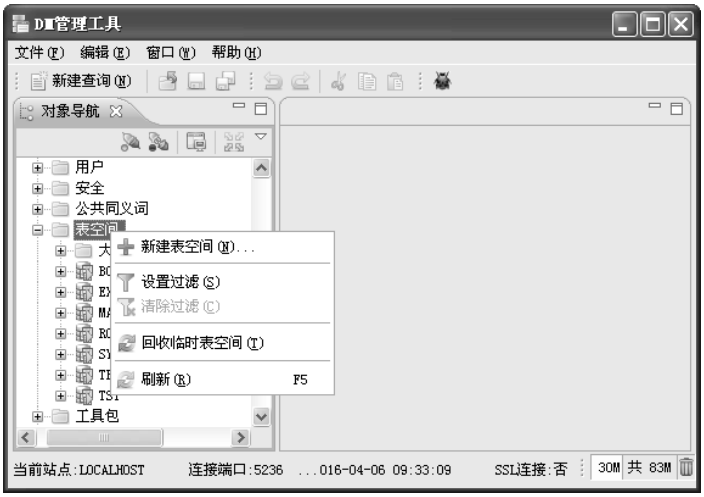

图 3-1　新建表空间

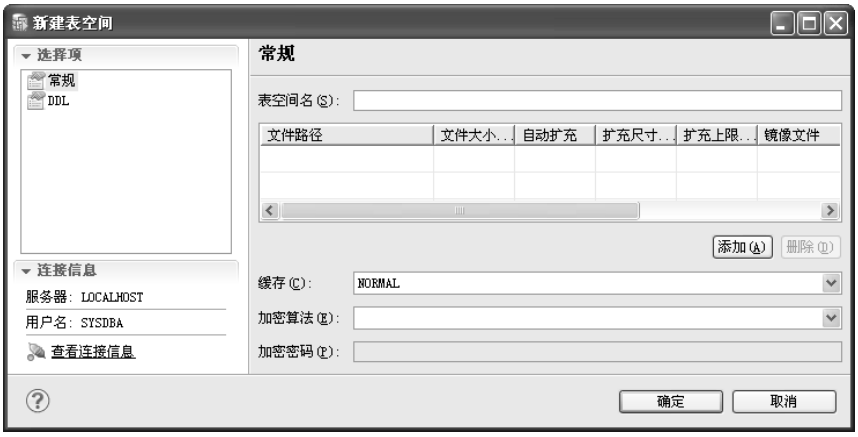

图 3-2　表空间参数设置界面

表 3-2　参数说明

参　数	说　　明
表空间名	表空间的名称
文件路径	数据文件的路径。可以单击浏览按钮浏览本地数据文件路径，也可以手动输入数据文件路径，但该路径应该对服务器端有效，否则无法创建
文件大小	数据文件的大小，单位为 MB
自动扩充	数据文件的自动扩充属性状态，包括以下三种情况。 默认：指使用服务器默认设置； 打开：指开启数据文件的自动扩充； 关闭：指关闭数据文件的自动扩充
扩充尺寸	数据文件每次扩展的大小，单位为 MB
扩充上限	数据文件可以扩充到的最大值，单位为 MB

步骤 3：在图 3-2 中单击“添加”按钮，在表格中自动添加一个新行，数据文件大小默认为 32，修改为 128，在文件路径单元格中输入或选择“C:\dmdbms\data\DAMENG\EXAMPLE2.dbf”文件。其他参数不变，结果如图 3-3 所示。

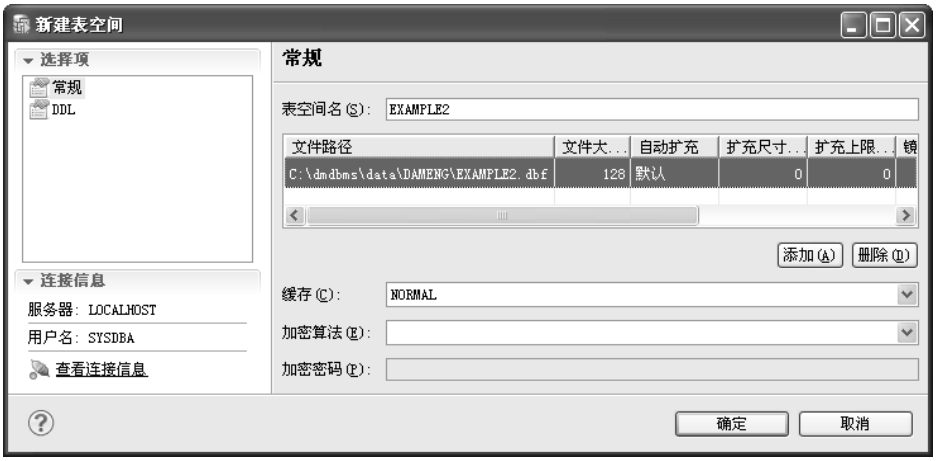

图 3-3　新建 EXAMPLE2 表空间

步骤 4：在图 3-3 中，单击“确定”按钮，完成 EXAMPLE2 表空间的创建过程。

3.2　修改表空间

随着数据库的数据量不断增加，原来创建的表空间不能满足数据存储的需要，应当适时对表空间进行修改，增加数据文件或者扩展数据文件的大小。我们既可以运用 SQL 命令来修改表空间，也可以通过 DM 管理工具来修改表空间，下面分别进行介绍。

3.2.1　用 SQL 命令修改表空间

1. 语法格式

修改表空间的 SQL 命令格式如下：

```
ALTER TABLESPACE <表空间名> [ONLINE | OFFLINE | <表空间重命名子句> | <数据文件重命名子句>|<增加数据文件子句>|<修改文件大小子句>|<修改文件自动扩展子句>|<数据页缓冲池子句>];
```

其中，各子句说明如下：

```
<表空间重命名子句> ::= RENAME TO <表空间名>
<数据文件重命名子句>::= RENAME DATAFILE <文件路径>{,<文件路径>} TO <文件路径>{,<文件路径>}
<增加数据文件子句> ::= ADD <数据文件子句>
<修改文件大小子句> ::= RESIZE DATAFILE <文件路径> TO <文件大小>
<修改文件自动扩展子句> ::= DATAFILE <文件路径>{,<文件路径>}[<自动扩展子句>]
```

通过这条命令，可以设置表空间脱机或联机，可以修改表空间的名称，可以修改数据文件的名称，可以增加数据文件，可以修改数据文件大小，还可以修改数据文件的自动扩展特性等。

2. 应用举例

【例 3-5】给 TS1 表空间增加数据文件 TS103.DBF，大小为 128MB。

```
SQL>ALTER TABLESPACE ts1 ADD DATAFILE 'C:\dmdbms\data\DAMENG\TS103.DBF' SIZE 128;
```

【例 3-6】修改 TS1 表空间数据文件 TS103.DBF 的大小为 256MB。

```
SQL>ALTER TABLESPACE ts1 RESIZE DATAFILE 'C:\dmdbms\data\DAMENG\TS103.DBF' TO 256;
```

【例 3-7】重命名数据文件 TS103.DBF 为 TS_103.DBF。

重命名数据文件时必须先将数据文件设置为离线状态，然后才能重命名文件。

（1）设置数据文件离线：

```
SQL>ALTER TABLESPACE ts1 OFFLINE;
```

（2）修改数据文件名：

```
SQL>ALTER TABLESPACE ts1 RENAME DATAFILE 'C:\dmdbms\data\DAMENG\TS103.DBF' TO 'C:\dmdbms\data\DAMENG\TS_103.DBF';
```

（3）设置数据文件在线：

```
SQL>ALTER TABLESPACE ts1 ONLINE;
```

【例 3-8】修改数据文件 TS102.DBF 为自动增长，每次增长 4MB，最大 1024MB。

```
SQL>ALTER TABLESPACE ts1 DATAFILE 'C:\dmdbms\data\DAMENG\TS102.DBF' AUTOEXTEND ON NEXT 4 MAXSIZE 1024;
```

【例 3-9】将 TS1 表空间重命名为 TS_1。

```
SQL>ALTER TABLESPACE ts1 RENAME TO ts_1;
```

【例 3-10】修改 TS_1 表空间缓冲区类别为 KEEP。

```
SQL>ALTER TABLESPACE ts_1 CACHE="KEEP";
```

注意，KEEP 要大写并加上双引号。

3. 附加说明

（1）修改表空间的用户必须具有 DBA 权限。

（2）修改表空间数据文件大小时，其大小必须大于自身大小。

（3）如果表空间有未提交事务，表空间不能修改 OFFLINE 状态。

（4）重命名表空间数据文件时，表空间必须处于 OFFLINE 状态，修改成功后再将表空间修改为 ONLINE 状态。

3.2.2 用管理工具修改表空间

【例 3-11】将 TS_1 表空间名称改回 TS1，缓冲池修改为 NORMAL，将数据文件 TS_103.DBF 修改为 TS103.DBF，初始大小为改为 768MB，数据文件 TS102.DBF 修改为不

能自动扩展。

步骤 1：在图 3-4 中右击 TS_1 表空间，在弹出的快捷菜单中选择“重命名”选项。在弹出的对话框中，输入新名称“TS1”，如图 3-5 所示，单击“确定”按钮。

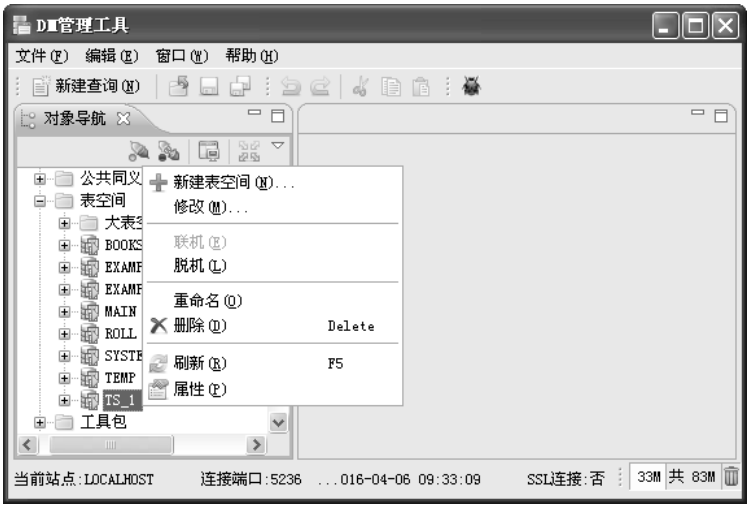

图 3-4　重命名表空间

图 3-5　输入表空间名称

步骤 2：右击 TS1 表空间，在弹出的快捷菜单中选择“修改”选项，弹出修改表空间对话框，如图 3-6 所示。

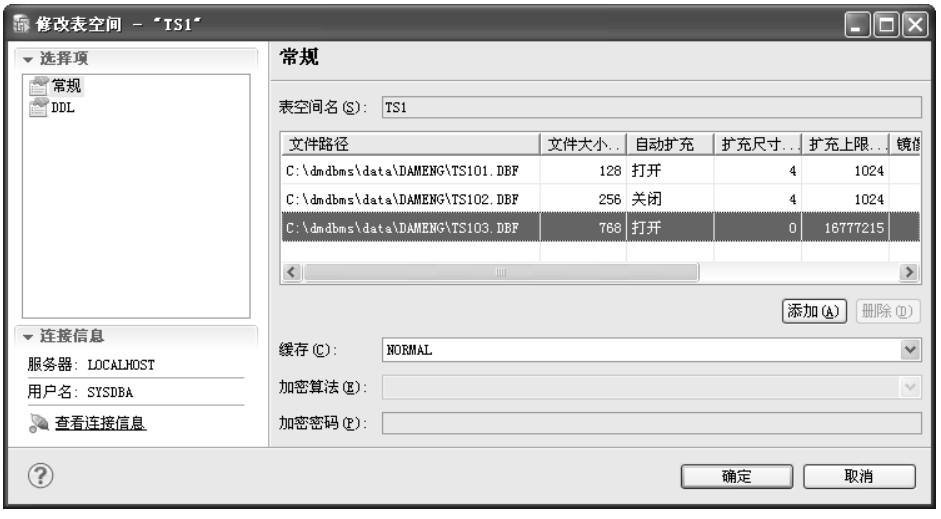

图 3-6　修改表空间参数

步骤 3：在图 3-6 中，选中数据文件“C:\dmdbms\data\DAMENG\TS_103.DBF”，将文件路径修改为“C:\dmdbms\data\DAMENG\TS103.DBF”，将文件大小修改为 768。

步骤 4：在图 3-6 中，选中数据文件“C:\dmdbms\data\DAMENG\TS102.DBF”，将自动扩充修改为“ 关闭”。单击“缓存”下拉按钮，选择“NORMAL”选项。

步骤 5：在图 3-6 中，单击“确定”按钮，完成表空间修改过程。

3.3 删除表空间

虽然实际工作中很少进行删除表空间的操作，但是掌握删除表空间的方法还是有必要的。由于表空间中存储了表、视图、索引等数据对象，删除表空间必然带来数据损失，所以达梦数据库对删除表空间有严格限制。我们既可以运用 SQL 命令来删除表空间，也可以通过 DM 管理工具来删除表空间，下面分别进行介绍。

3.3.1 用 SQL 命令删除表空间

1. 语法格式

删除表空间的 SQL 命令格式如下：

```
DROP TABLESPACE〈表空间名〉
```

2. 应用举例

【例 3-12】删除表空间 TS2。

```
SQL>DROP TABLESPACE ts2;
```

如果 TS2 表空间存在数据，则不能直接删除表空间。

【例 3-13】试图删除 TEMP 表空间。

```
SQL>DROP TABLESPACE temp;
DROP TABLESPACE temp;
第 1 行附近出现错误[-3418]:系统表空间[TEMP]不能被删除.
```

这个例子说明删除表空间是有限制的，数据库安装过程中系统创建的表空间 SYSTEM、TEMP 等不允许删除。如果表空间中已经存在数据对象，则该表空间也不允许删除。

3. 附加说明

（1）SYSTEM、RLOG、ROLL 和 TEMP 表空间不允许删除。

（2）用该语句的用户必须具有 DBA 权限。

（3）系统处于 SUSPEND 或 MOUNT 状态时不允许删除表空间，系统只有处于 OPEN 状态下才允许删除表空间。

（4）如果表空间存放了数据，则不允许删除表空间。如果确实要删除表空间，则必须先删除表空间中的数据对象。

3.3.2　用管理工具删除表空间

【例 3-14】删除表空间 EXAMPLE2。

步骤 1：在图 3-7 中右击表空间 EXAMPLE2，在弹出的快捷菜单中选择“删除”选项，进入删除表空间主界面，如图 3-8 所示。

步骤 2：在图 3-8 中列出了被删除表空间的对象名、对象类型、执行状态、反馈消息等内容。EXAMPLE2 处于等待删除的状态，“取消”按钮表示不删除，“确定”按钮表示删除。单击“确定”按钮后，完成 EXAMPLE2 表空间及其数据文件的删除。

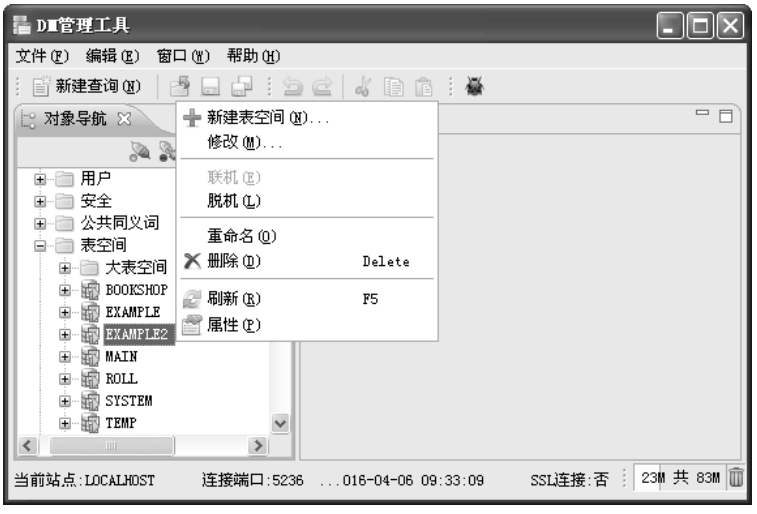

图 3-7　删除表空间

图 3-8　删除表空间主界面

3.4 创建大表空间

大表空间即 HTS（Huge Table Space）。HTS 表空间是专门用来存放 HFS（Huge File System）表的，如果不创建 HTS 表空间，就只能使用系统表空间 HMAIN。为了尽量少使用系统表空间 HMAIN，通常需要另外创建大表空间。

HTS 表空间 ID 取值为 0～32767，ID 由系统自动分配，ID 不能重复使用，即使删除已有 HTS 表空间，也无法重复使用已用 ID，也就是说，创建 32768 次 HTS 表空间后，就无法再创建 HTS 表空间了。这个表空间与普通的表空间不同。普通的表空间，数据是通过段、簇、页来管理的，并且以固定大小（4KB、8KB、16KB、32KB）的页面为管理单位；而 HTS 相当于一个简单的文件系统，创建一个 HTS，其实就是创建一个空的目录（系统中有一个默认 HTS，目录名为 HMAIN）。HTS 表空间的相关信息存储在动态视图 v$huge_tablespace 中。

可以用 SQL 命令和 DM 管理工具两种方式创建 HTS 表空间，下面分别进行介绍。

3.4.1 用 SQL 命令创建大表空间

1. 语法格式

创建大表空间的 SQL 命令格式如下：

```
SQL>CREATE HUGE TABLESPACE <表空间名> PATH <表空间路径>;
```

语法格式中，<表空间路径>是指新创建的表空间在操作系统下的路径。

2. 应用举例

【例 3-15】创建两个大表空间，一个名称为 HTS0，路径为 C:\ dmdbms\data\DAMENG\HTS0；另一个名称为 HTS1，路径为 C:\ dmdbms\data\DAMENG\HTS1。

```
SQL>CREATE HUGE TABLESPACE hts0 PATH 'C:\dmdbms\data\DAMENG\HTS0';
SQL>CREATE HUGE TABLESPACE hts1 PATH 'C:\dmdbms\data\DAMENG\HTS1';
```

上述命令中，HTS0 目录和 HTS1 目录不能事先存在，如果存在则无法创建 HTS0 大表空间。

3. 附加说明

（1）表空间名在服务器中必须唯一。

（2）创建大表空间的用户必须具有 DBA 权限。

3.4.2 用管理工具创建大表空间

【例 3-16】创建一个大表空间，名称为 HTS2，路径为 C:\dbdbms\ data\DAMENG\HTS2。

步骤 1：在图 3-9 中右击“大表空间”节点，在弹出的快捷菜单中选择“新建大表空间”选项，弹出“新建大表空间”对话框，如图 3-10 所示。

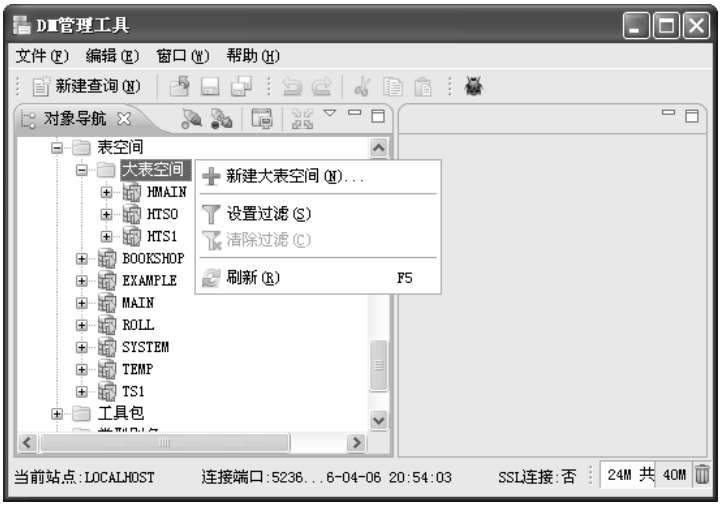

图 3-9　新建大表空间

图 3-10　设置大表空间参数

步骤 2：在图 3-10 中，在“表空间名”文本框中填写“HTS2”，在“路径”文本框中选择“C:\dbdbms\data\DAMENG\HTS2”。

步骤 3：单击“确定”按钮，完成大表空间创建过程。

3.5　删除大表空间

如果大表空间没有被使用，那么这个大表空间可以被删除。即使很少要删除大表空间，但是掌握删除大表空间的方法仍然是很有必要的，这里介绍使用 SQL 命令和 DM 管理工具删除大表空间的方法。

3.5.1 用 SQL 命令删除大表空间

1. 语法格式

删除大表空间的 SQL 命令格式如下。

```
DROP HUGE TABLESPACE <表空间名>
```

2. 应用举例

【例 3-17】以 SYSDBA 身份登录数据库后，删除大表空间 HTS0。

```
SQL>DROP HUGE TABLESPACE HTS0;
```

这个例子说明，只要表空间中没有数据，可以直接删除表空间。

3. 附加说明

（1）删除大表空间的用户必须具有 DBA 权限。
（2）大表空间中没有数据才能被删除。

3.5.2 用管理工具删除大表空间

【例 3-18】删除大表空间 HTS2。

步骤 1：在图 3-11 中右击 HTS2 大表空间，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 3-12 所示。

步骤 2：在图 3-12 中，确认无误后，单击“确定”按钮，完成删除 HTS2 大表空间的过程。

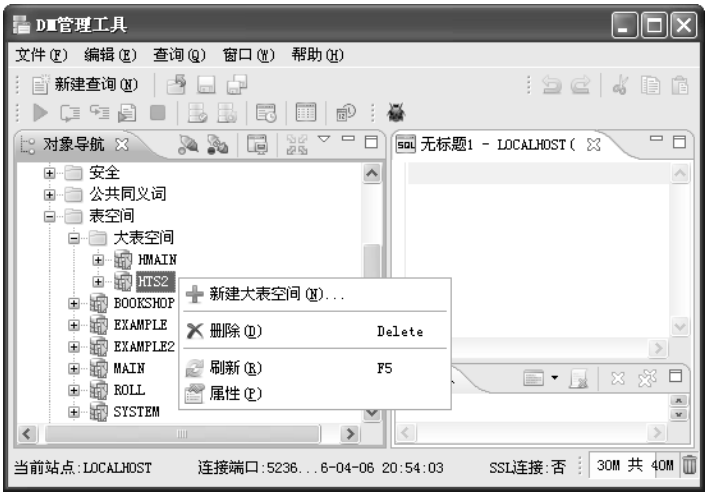

图 3-11 删除大表空间

图 3-12　确认删除大表空间

4

第 4 章

对象管理

如果把数据库比作仓库，那么表空间只相当于一间库房，库房中还缺少用来搁置各类货物的货架，也就是说仅有表空间还不能存储数据，需要建立各种各样的表才能存储数据。创建表之前还要先建立用户和模式，为了查询数据更加快捷，通常还要在表的字段上建立索引，有时还要对一个或多个表建立视图。这里所提到的用户、模式、表、索引、视图，以及序列、同义词等都是 DM 数据库对象。

4.1 用户管理

用户在数据库中具有双重意义，从安全角度看，只有在数据库内部建立用户以后，使用者才能以该用户名登录和使用数据库，从管理角度看，数据库的对象（除全局同义词外）都是按照用户的模式名来进行组织管理的，或者说数据库对象都被组织到用户的模式下，因此用户管理是 DM 对象管理的基础。

4.1.1 创建用户

创建用户时要指定相关的要素，通常包括用户登录数据库时的身份验证模式与口令、用户拥有对象存储的默认表空间、对用户访问数据库资源的各项限制等。我们可以通过 SQL 命令和 DM 管理工具来创建用户。

1. 用 SQL 命令创建用户

1）语法格式

创建用户的 SQL 命令格式如下：

```
CREATE USER <用户名> IDENTIFIED <身份验证模式>  [PASSWORD_POLICY <口令策略>][<
锁定子句>][<存储加密密钥>][<空间限制子句>][<只读标志>][<资源限制子句>][<允许 IP 子句>][<
禁止 IP 子句>][<允许时间子句>][<禁止时间子句>][< TABLESPACE 子句>]
```

其中，各子句说明如下：

```
<身份验证模式> ::= <数据库身份验证模式>|<外部身份验证模式>
<数据库身份验证模式> ::= BY <口令>
<外部身份验证模式> ::= EXTERNALLY
<口令策略> ::= 口令策略项的任意组合
<锁定子句> ::= ACCOUNT LOCK | ACCOUNT UNLOCK
<存储加密密钥> ::= ENCRYPT BY <口令>
<空间限制子句> ::= DISKSPACE LIMIT <空间大小>| DISKSPACE UNLIMITED
<只读标志> ::= READ ONLY | NOT READ ONLY
<资源限制子句> ::= LIMIT <资源设置项>{,<资源设置项>}
<资源设置项> ::= SESSION_PER_USER <参数设置>|
        CONNECT_IDLE_TIME <参数设置>|
        CONNECT_TIME <参数设置>|
        CPU_PER_CALL <参数设置>|
        CPU_PER_SESSION <参数设置>|
        MEM_SPACE <参数设置>|
        READ_PER_CALL <参数设置>|
        READ_PER_SESSION <参数设置>|
        FAILED_LOGIN_ATTEMPS <参数设置>|
        PASSWORD_LIFE_TIME <参数设置>|
        PASSWORD_REUSE_TIME <参数设置>|
        PASSWORD_REUSE_MAX <参数设置>|
        PASSWORD_LOCK_TIME <参数设置>|
        PASSWORD_GRACE_TIME <参数设置>
<参数设置> ::=<参数值>| UNLIMITED
<允许 IP 子句> ::= ALLOW_IP <IP 项>{,<IP 项>}
<禁止 IP 子句> ::= NOT_ALLOW_IP <IP 项>{,<IP 项>}
<IP 项> ::= <具体 IP>|<网段>
<允许时间子句> ::= ALLOW_DATETIME <时间项>{,<时间项>}
<禁止时间子句> ::= NOT_ALLOW_DATETIME <时间项>{,<时间项>}
<时间项> ::= <具体时间段> | <规则时间段>
<具体时间段> ::= <具体日期> <具体时间> TO <具体日期> <具体时间>
<规则时间段> ::= <规则时间标志> <具体时间> TO <规则时间标志> <具体时间>
<规则时间标志> ::= MON | TUE | WED | THURS | FRI | SAT | SUN
<TABLESPACE 子句> ::=DEFAULT TABLESPACE <表空间名>
```

语法格式中的参数说明见表 4-1 和表 4-2。

表 4-1　参数说明

参　数	说　　明
用户名	用户名称最大长度为 128 字节
身份验证模式	<数据库身份验证模式>格式为 IDENTIFIED BY 密码 <外部身份验证模式>格式为 IDENTIFIED EXTERNALLY 基于 OS 的身份验证分为本机验证和远程验证，本机验证在任何情况下都可以使用，而远程验证则需要将配置文件 dm.ini 的 ENABLE_REMOTE_ OSAUTH 项设置为 1(默认为 0)，表示支持远程验证，同时要将配置文件 dm.ini 的 ENABLE_ENCRYPT 项设置为 1，表示采用 SSL 安全连接
口令策略	可以为以下值，或其任何组合。 0：　表示无策略。 1：　表示禁止与用户名相同。 2：　表示口令长度不小于 6。 4：　表示至少包含一个大写字母（A～Z）。 8：　表示至少包含一个数字（0～9）。 16：　表示至少包含一个标点符号。 其他：表示以上设置值的和，如 3＝1+2，表示同时启用第 1 项和第 2 项策略。当设置为 0 时，表示设置口令没有限制，但总长度不得超过 48 个字节。另外，若不指定该项，则默认采用系统配置文件中 PWD_POLICY 所设的值
存储加密密钥	存储加密密钥用于与半透明加密配合使用，默认情况下系统自动生成一个密钥
只读标志	只读标志表示该登录是否只能对数据库做只读操作，默认为可读写
资源限制子句	资源设置项的各参数设置说明见表 4-2
允许 IP 子句 禁止 IP 子句	允许 IP 和禁止 IP 用于控制此登录是否可以从某个 IP 访问数据库，其中禁止 IP 优先。在设置 IP 时，可以利用*来设置网段，如 192.168.0.*
允许时间子句 禁止时间子句	允许时间段和禁止时间段用于控制此登录是否可以在某个时间段访问数据库，其中禁止时间段优先。在设置时间段时，有以下两种方式。 （1）具体时间段，如 2006 年 1 月 1 日 8:30 至 2006 年 2 月 1 日 17:00。 （2）规则时间段，如每周一 8:30 至每周五 17:00。 口令策略、允许 IP、禁止 IP、允许时间段、禁止时间段和外部身份验证功能只在安全版本中提供
TABLESPACE 子句	TABLESPACE 子句格式为 DEFAULT TABLESPACE <表空间名>。 用户默认表空间不能使用 RLOG、ROLL、TEMP 表空间

表 4-2　资源设置项说明

资源设置项	说　　明	最大值	最小值	默认值
SESSION_PER_USER	在一个实例中，一个用户可以同时拥有的会话数量	32768	1	系统所能提供的最大值
CONNECT_TIME	一个会话连接、访问和操作数据库服务器的时间上限（单位：10 分钟）	144（1 天）	1	无限制

（续表）

资源设置项	说　　明	最大值	最小值	默认值
CONNECT_IDLE_TIME	会话最大空闲时间（单位：10 分钟）	144（1 天）	1	无限制
FAILED_LOGIN_ATTEMPS	将引起一个账户被锁定的连续注册失败的次数	100	1	3
CPU_PER_SESSION	一个会话允许使用的 CPU 时间上限（单位：秒）	31536000（365 天）	1	无限制
CPU_PER_CALL	用户的一个请求能够使用的 CPU 时间上限（单位：秒）	86400（1 天）	1	无限制
READ_PER_SESSION	会话能够读取的总数据页上限	2147483646	1	无限制
READ_PER_CALL	每个请求能够读取的数据页数	2147483646	1	无限制
MEM_SPACE	会话占有的私有内存空间上限（单位：MB）	2147483647	1	无限制
PASSWORD_LIFE_TIME	一个口令在其终止前可以使用的天数	365	1	无限制
PASSWORD_REUSE_TIME	一个口令在可以重新使用前必须经过的天数	365	1	无限制
PASSWORD_REUSE_MAX	一个口令在可以重新使用前必须改变的次数	32768	1	无限制
PASSWORD_LOCK_TIME	如果超过 FAILED_ LOGIN_ ATTEMPS 设置值，一个账户将被锁定的分钟数	1440（1 天）	1	1
PASSWORD_GRACE_TIME	以天为单位的口令过期宽限时间	30	1	10

2）应用举例

【例 4-1】创建 USER0 用户，口令为 pworduser0。

```
SQL>CONN SYSDBA/SYSDBA;
SQL>CREATE USER user0 IDENTIFIED BY "pworduser0";
```

由于这个例子的口令中要求小写字母，所以口令要加双引号，如果不加双引号，口令自动转换为大写。

【例 4-2】创建 USER1 用户，口令为 PWORDUSER1，会话超时为 3 分钟，默认表空间为 EXAMPLE。

```
SQL>CREATE USER user1 IDENTIFIED BY pworduser1 LIMIT CONNECT_TIME 3 DEFAULT TABLESPACE example;
```

这个例子中口令 pworduser1 虽然使用了小写，在库中实际上它是大写的，在使用 user1 或 USER1 登录时，口令一定要用大写，即 PWORDUSER1。如果确实需要使用小写或大小写混合密码，则密码一定要加西文双引号。注意，表空间子句放在最后面。

【例 4-3】创建 USER2 用户，口令为 PWORDUSER2，默认表空间同样为 EXAMPLE。

```
SQL>CREATE USER user2 IDENTIFIED BY pworduser2 DEFAULT TABLESPACE example;
```

这个例子说明不同用户可以使用同样的表空间。

3）附加说明

（1）用户名在服务器中必须唯一。

（2）用户口令以密文形式存储。

（3）系统预先设置了三个用户，分别为 SYSDBA、SYSAUDITOR 和 SYSSSO，其中 SYSDBA 具备 DBA 角色，SYSAUDITOR 具备 DB_AUDIT_ ADMIN 角色，而 SYSSSO 具备 DB_POLICY_ADMIN 系统角色。

（4）DM 提供三种身份验证模式来保护服务器访问的安全，即数据库身份验证模式、外部身份验证模式和混合身份验证模式。数据库身份验证模式需要利用数据库口令；外部身份验证模式既支持基于操作系统的身份验证又提供了口令管理策略；混合身份验证模式同时支持数字证书和数据库身份的双重验证。

2．用管理工具创建用户

【例 4-4】创建 USER3 用户，采用密码验证方式，口令为 DMUSER123，使用表空间为 TS1，会话持续期（会话超时）为 30 分钟。

步骤 1：在图 4-1 中右击“管理用户”节点，在弹出的快捷菜单中选择“新建用户”选项，弹出“新建用户”对话框，如图 4-2 所示。

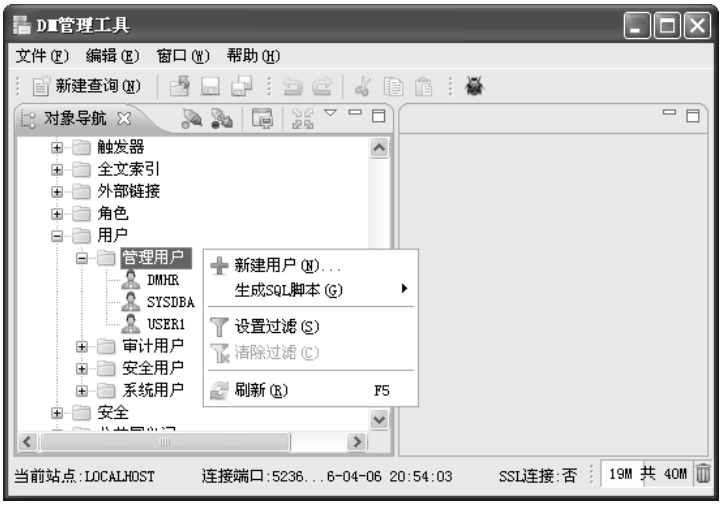

图 4-1　新建用户

步骤 2：在图 4-2 中，用户名设置为“USER3”，连接验证方式选择“密码验证”，密码为“DMUSER123”，表空间为“TS1”。

步骤 3：进入“资源限制”页面，如图 4-3 所示，选中“会话持续期”的“限制”复选框，限制为 3 分钟。其他参数忽略，默认即可。

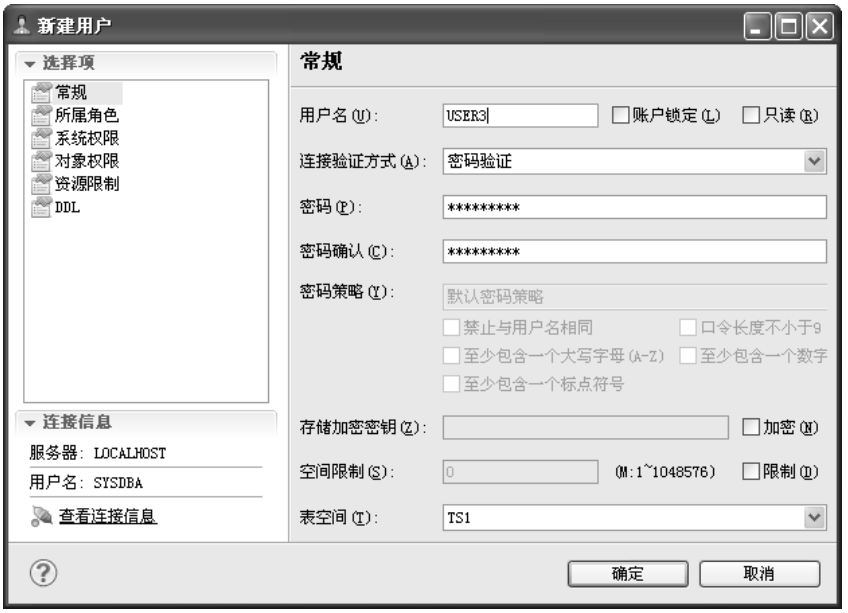

图 4-2　设置常规参数

图 4-3　设置资源限制参数

步骤 4：单击“确定”按钮，完成新建用户过程。

4.1.2　修改用户

在数据库使用过程中，我们可能需要修改用户的登录密码，修改用户默认使用的表空间，等等，这些都是修改用户的操作。可以采用 SQL 命令或 DM 管理工具来修改用户。

1．用 SQL 命令修改用户

1）语法格式

修改用户的 SQL 命令格式如下：

```
ALTER USER <用户名> [IDENTIFIED <身份验证模式>] [PASSWORD_POLICY <口令策略>] [<锁
```

```
定子句>] [<存储加密密钥>] [<空间限制子句>] [<只读标志>][<资源限制子句>][<允许 IP 子句>][<禁止 IP 子句>][<允许时间子句>][<禁止时间子句>][< TABLESPACE 子句>]
```

其中，各子句的格式和参数说明同创建用户的 SQL 命令。

2）应用举例

【例 4-5】将 DMHR 用户的密码修改为 DMHR12345。

```
SQL>CONN SYSDBA/SYSDBA;
SQL>ALTER USER DMHR IDENTIFIED BY DMHR12345;
```

这个例子的前提条件是数据库中已经存在 DMHR 用户。如果忘记了某个用户的密码，用这种方法可以重新设置密码。

3）附加说明

（1）每个用户均可修改自身的口令，SYSDBA 用户可强制修改非系统预设用户的口令（在数据库验证方式下）。

（2）只有具备 ALTER USER 权限的用户才能修改其身份验证模式、系统角色及资源限制项。

（3）不论 dm.ini 的 DDL_AUTO_COMMIT 设置为自动提交还是非自动提交，ALTER USER 操作都会被自动提交。

（4）系统预设用户不能修改其系统角色和资源限制项。

2．用管理工具修改用户

【例 4-6】修改用户 USER3，口令为 USER33333，口令登录错误 5 次就锁定账户。

步骤 1：在图 4-4 中右击用户 USER3，在弹出的快捷菜单中选择“修改”选项，弹出修改用户对话框，如图 4-5 所示。

图 4-4　修改用户

步骤 2：在图 4-5 中，将“密码”和“密码确认”修改为 USER33333。

步骤 3：进入“资源限制”页面，如图 4-6 所示，设置登录失败次数为 5。

步骤 4：在图 4-6 中，单击“确定”按钮，完成修改用户过程。

图 4-5　修改用户密码

图 4-6　修改登录失败次数

4.1.3　删除用户

前面已经讲过，数据库的对象都是组织在用户的模式下的，如果删除用户，那么用户模式下的数据对象都要删除，并且对该用户模式下的数据对象的依赖也都会被删除。删除用户应当慎重，可以使用 SQL 命令或 DM 管理工具来删除用户。

1．用 SQL 命令删除用户

1）语法格式

删除用户的 SQL 命令格式如下：

```
DROP USER <用户名> [RESTRICT | CASCADE];
```

用户名指要删除的用户名称。

2）应用举例

【例 4-7】以用户 SYSDBA 登录，删除用户 USER0。

```
SQL>DROP USER user0;
```

如果用户 USER0 存在数据对象，则不能删除该用户。

【例 4-8】以用户 SYSDBA 登录，尝试删除用户 SYSSSO。

```
SQL>DROP USER syssso;
```

```
DROP USER syssso;
第 1 行附近出现错误[-5533]:没有删除用户权限.
```

这个例子说明，不能删除系统自动创建的 SYSSSO 用户。

3）附加说明

（1）系统自动创建的三个系统用户 SYSDBA、SYSAUDITOR 和 SYSSSO 不能被删除。

（2）具有相应的 DROP USER 权限的用户即可进行删除用户操作。

（3）删除用户会删除该用户建立的所有对象，且不可恢复。如果要保存这些实体，请参考 REVOKE 语句。

（4）如果未使用 CASCADE 选项，若该用户建立了数据库对象（如表、视图、过程或函数），或其他用户对象引用了该用户的对象，或在该用户的表上存在其他用户建立的视图，DM 将返回错误信息，而不删除此用户。

（5）如果使用了 CASCADE 选项，除数据库中该用户及其创建的所有对象被删除外，如果其他用户创建的表引用了该用户表上的主关键字或唯一关键字，或者在该表上创建了视图，DM 还将自动删除相应的参照完整性约束及视图依赖关系。

（6）正在使用的用户可以被删除，删除后重登录或者进行操作会报错。

2．用管理工具删除用户

【例 4-9】删除用户 USER3。

步骤 1：在图 4-7 中右击 USER3 用户，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-8 所示。

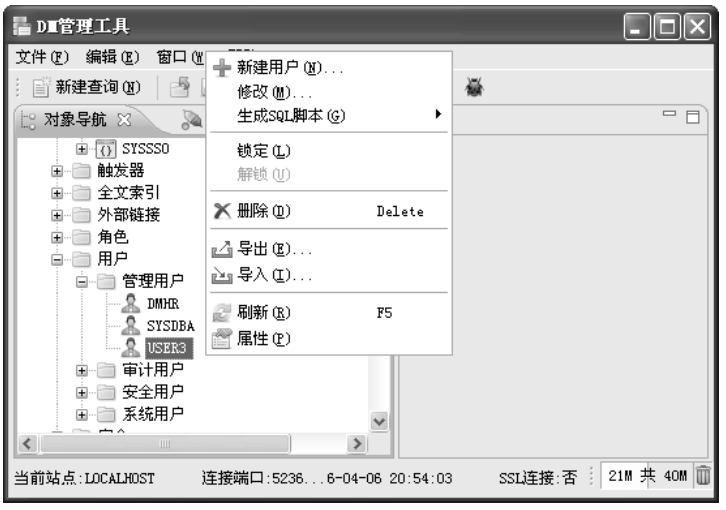

图 4-7　右击用户

步骤 2：在图 4-8 中，单击“确定”按钮，删除 USER3 用户。

图 4-8　删除用户

4.2　模式管理

在 DM 中，系统为每一个用户自动建立了一个与用户名同名的模式作为默认模式，用户还可以用模式定义语句建立其他模式。一个用户可以创建多个模式，一个模式只归属于一个用户，一个模式中的对象（表、视图等）可以被该用户使用，也可以授权给其他用户访问。

4.2.1　创建模式

创建模式时要指定归属的用户名，可以在创建模式的同时创建模式中的对象，但通常是分开进行的。

1. 用 SQL 命令创建模式

1）语法格式

创建模式的 SQL 命令格式如下：

```
<模式定义子句 1>|<模式定义子句 2>
```

其中，各子句说明如下：

```
<模式定义子句 1> ::= CREATE SCHEMA <模式名> [AUTHORIZATION <用户名>][<DDL_GRANT 子句> {<DDL_GRANT 子句>}];
<模式定义子句 2> ::= CREATE SCHEMA AUTHORIZATION <用户名> [<DDL_GRANT 子句> {<DDL_GRANT 子句>}]
<DDL_GRANT 子句> ::= <基表定义>|<域定义>|<基表修改>|<索引定义>|<视图定义>|<序列定义>|<存储过程定义>|<存储函数定义>|<触发器定义>|<特权定义>|<全文索引定义>|<同义词定义>|<包定义>|<包体定义>|<类定义>|<类体定义>|<外部链接定义>]|<物化视图定义>|<物化视图日志定义>|<注释定义>
```

<用户名>指明给哪个用户创建模式，如果省略用户名，则默认给当前用户创建模式。语法格式中其他部分都是可选项，如<表定义>、<表修改>、<视图定义>等子句，后面章节

将详细介绍。

2）应用举例

【例 4-10】以用户 SYSDBA 登录，为 DMHR 用户增加一个模式，模式名为 DMHR2，并在 DMHR2 模式中定义一张表 TAB1（下一节将具体介绍表的管理）。

```
SQL>CREATE SCHEMA dmhr2 AUTHORIZATION dmhr
CREATE TABLE tab1(id INT, name VARCHAR(20));
/
```

注意，该命令是连续的，第一行没有分号，该 SQL 命令最终不是以分号结束的，而以“/”结束。

3）附加说明

（1）<模式名>不可与其所在数据库中其他模式名相同；在创建新的模式时，如果存在同名的模式，那么该命令不能执行。

（2）使用该语句的用户必须具有 DBA 或 CREATE SCHEMA 权限。

（3）模式一旦定义，该用户所建基表、视图等均属于该模式，其他用户访问该用户所建立的基表、视图等均需在表名、视图名前冠以模式名；而建表者访问自己当前模式所建表、视图时模式名可省；若没有指定当前模式，则系统自动以当前用户名作为模式名。

（4）模式定义语句不允许与其他 SQL 语句一起执行。

（5）在 disql 中使用该语句必须以“/”结束。

2．用管理工具创建模式

【例 4-11】以用户 SYSDBA 给 DMHR 用户创建一个模式，名称为 DMHR3。

步骤 1：在图 4-9 中右击“模式”节点，在弹出的快捷菜单中选择“新建模式”选项，弹出“新建模式”对话框，如图 4-10 所示。

图 4-9　新建模式

图 4-10　设置模式名

步骤 2：在图 4-10 中进入常规参数页面，设置模式名为“DMHR3”。单击“选择用户”按钮，弹出“选择（用户）”对话框，如图 4-11 所示，选中 DMHR 用户并单击“确定”按钮返回。

图 4-11　选择用户

步骤 3：在图 4-10 中，单击“确定”按钮，完成模式创建过程。

4.2.2　设置当前模式

当一个用户有多个模式时，可以指定一个模式为当前默认模式，用 SQL 命令来设置当前模式。

1. 语法格式

设置当前模式的 SQL 命令格式如下：

```
SET SCHEMA <模式名>;
```

2. 应用举例

【例 4-12】将 DMHR2 模式设置为 DMHR 用户的当前模式，并在 DMHR2 模式下的 TAB1 表中增加一条记录。

（1）以 DMHR 用户登录：

```
SQL>CONN dmhr/DMHR12345;
服务器[LOCALHOST:5236]:处于普通打开状态
使用普通用户登录
```

（2）设置 DMHR 用户的当前模式为 DMHR2 ：

```
SQL>SET SCHEMA dmhr2;
```

（3）在 TAB1 表中插入一条记录：

```
SQL>INSERT INTO tab1 VALUES(1, '武汉');
SQL>COMMIT;
```

（4）查询 TAB1 表的数据：

```
SQL>SELECT * FROM tab1;
行号          ID          NAME
---------- ----------- -------------------------
1             1           武汉
```

DMHR 用户登录后，由于已经设置当前模式，表名 TAB1 之前不需要加模式名。

3. 附加说明

一个用户只能设置自己的某个模式为该用户的当前模式。

4.2.3 删除模式

在 DM 系统中，允许用户删除整个模式，当模式下有表或视图等数据库对象时，必须采取级联删除，否则删除失败。可以通过 SQL 命令或 DM 管理工具来删除模式。

1. 用 SQL 命令删除模式

1）语法格式

删除模式的 SQL 命令格式如下：

```
DROP SCHEMA <模式名> [RESTRICT | CASCADE];
```

如果使用 RESTRICT 选项，只有当模式为空时删除才能成功，否则，当模式中存在数据库对象时则删除失败。默认选项为 RESTRICT 选项。

如果使用 CASCADE 选项，则整个模式、模式中的对象，以及与该模式相关的依赖关系都被删除。

2）应用举例

【例 4-13】以用户 SYSDBA 登录，删除 DMHR2 模式。

（1）以 SYSDBA 用户登录数据库：

```
SQL> CONN SYSDBA/SYSDBA;
服务器[LOCALHOST:5236]:处于普通打开状态
使用普通用户登录
```

（2）直接删除 DMHR2 模式：

```
SQL> DROP SCHEMA dmhr2;
DROP SCHEMA dmhr2;
第 1 行附近出现错误[-5001]:模式[DMHR2]不为空.
```

删除失败的原因是 DMHR2 模式不为空，存在数据库对象 TAB1，不能删除非空的模式。

（3）使用 CASCADE 选项删除 DMHR2 模式：

```
SQL> DROP SCHEMA dmhr2 CASCADE;
```

该命令执行成功，使用 CASCADE 选项将整个模式、模式中的对象及其依赖关系全部删除。

3）附加说明

（1）<模式名>必须是当前数据库中已经存在的模式。

（2）用该语句的用户必须具有 DBA 权限或是该模式的所有者。

2. 用管理工具删除模式

【例 4-14】以 SYSDBA 用户登录，删除 DMHR3 模式。

步骤 1：在图 4-12 中右击 DMHR3 模式，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-13 所示。

图 4-12　删除模式

图 4-13　确认删除

步骤 2：在图 4-13 中，确认无误后，单击“确定”按钮，完成 DMHR3 模式的删除过程。

4.3 表管理

表是数据库中数据存储的基本单元，是对用户数据进行读和操纵的逻辑实体。表由列和行组成，每一行代表一个单独的记录。表中包含一组固定的列，表中的列描述该表所跟踪的实体的属性，每个列都有一个名称及特性。列的特性由两部分组成：数据类型和长度。对于 NUMERIC、DECIMAL 以及那些包含秒的时间间隔类型来说，可以指定列的小数位及精度特性。在 DM 系统中，CHAR、CHARACTER、VARCHAR 数据类型的最大长度由数据库页面的大小决定，数据库页面大小在初始化数据库时指定。

为了确保数据库中数据的一致性和完整性，在创建表时可以定义表的实体完整性、域完整性和参照完整性。实体完整性定义表中的所有行能唯一地标识，一般用主键、唯一索引、UNIQUE 关键字及 IDENTITY 属性来定义；域完整性通常指数据的有效性，限制数据类型、默认值、规则、约束、是否可以为空等条件，域完整性可以确保不会输入无效的值；参考完整性维护表间数据的有效性、完整性，通常通过建立外键对应另一表的主键来实现。

如果用户在创建表时没有定义表的完整性和一致性约束条件，那么用户可以利用 DM 所提供的表修改语句来进行补充或修改。DM 系统提供表修改语句，可对表的结构进行全面的修改，包括修改表名、列名、增加字段、删除字段、修改字段类型、增加表级约束、删除表级约束、设置字段默认值、设置触发器状态等一系列修改功能。

在达梦数据库中，表可以分为两类，即数据库表和外部表，数据库表由数据库管理系统自行组织管理，而外部表在数据库的外部组织，是操作系统文件。

这里分别介绍数据库表的创建、修改和删除，外部表的创建和删除。

4.3.1 管理数据库表

1．创建数据库表

1）用 SQL 命令创建数据库表

（1）语法格式：表结构的完整语法格式篇幅很长，为了便于读者学习，这里做一些必要的简化，详细内容列入到附录 A 中。创建数据库表的 SQL 命令格式如下：

```
CREATE [[GLOBAL] TEMPORARY] TABLE <表名定义> <表结构定义>;
```

其中，各子句简化说明如下：

```
<表名定义> ::= [<模式名>.] <表名>
<表结构定义> ::= (<字段定义> {,<字段定义>} [<表级约束定义>{,<表级约束定义>}])[<PARTITION 子句>][<空间限制子句>] [<STORAGE 子句>]
<字段定义>: :=<字段名> <字段类型> [DEFAULT <列默认值表达式>][<列级约束定义>]
```

〈列级约束定义〉::=[CONSTRAINT <约束名>] [NOT] NULL |<唯一性约束选项>|<引用约束>|[CHECK (<检验条件>)]

〈唯一性约束选项〉::=[PRIMARY KEY]|[[NOT] CLUSTER PRIMARY KEY]| [CLUSTER[UNIQUE] KEY]|UNIQUE

〈引用约束〉::=REFERENCES [<模式名>.]<表名>[(<列名>{[,<列名>]})]

〈表级约束定义〉::=[CONSTRAINT <约束名>]<唯一性约束选项>(<列名> {,<列名>})|FOREIGN KEY (<列名>{,<列名>}) <引用约束>|CHECK (<检验条件>)

表结构定义的核心是字段名和字段类型，还包括字段约束和表约束等，初学者掌握这几项即可。达梦数据库表还包括分区表、HFS 表、LIST 表等，在创建这些高性能表时，还需要指定专门的关键词和子句，在后续章节中将详细介绍。

（2）应用举例。

【例 4-15】在 DMHR 模式下创建 CLASSES、STUDENTS 和 STUDY 表，表的字段要求见表 4-3～表 4-6。

表 4-3　CLASSES 表

字段名	字段类型	主键	非空	唯一
CLASSID	NUMBER(4, 0)	是	是	是
CLASSNAME	VARCHAR(20)		是	
STARTDATE	DATE		是	
ENDDATE	DATE		是	

表 4-4　STUDENTS 表

字段名	字段类型	主键	非空	唯一
STUDENTID	VARCHAR(10)	是	是	是
NAME	VARCHAR(20)		是	
BIRTHDAY	DATE		是	

表 4-5　STUDY 表

字段名	字段类型	主键	非空	唯一
STUDENTID	VARCHAR(10)	是	是	是
CLASSID	NUMBER(4,0)	是	是	
STUDYDATE	DATE		是	

表 4-6　STUDY 表的外键设置

字段名	外键	参考
STUDENTID	是	STUDENTS(STUDENTID)
CLASSID	是	CLASSES(CLASSID)

① 创建 CLASSES 表：

```
SQL> CREATE TABLE dmhr.classes
(
classid NUMBER(4,0) PRIMARY KEY,
classname VARCHAR(20) NOT NULL,
startdate DATE NOT NULL,
enddate DATE NOT NULL
);
```

② 创建 STUDENTS 表：

```
SQL> CREATE TABLE dmhr.students
(
studentid VARCHAR(10) PRIMARY KEY,
name VARCHAR(20) NOT NULL,
birthday DATE NOT NULL
);
```

③ 创建 STUDY 表：

```
CREATE TABLE dmhr.study
(
studentid VARCHAR(10),
classid NUMBER(4,0),
studydate DATE NOT NULL,
CONSTRAINT studentid_fk FOREIGN KEY (studentid) REFERENCES dmhr.students(studentid),
CONSTRAINT classid_fk FOREIGN KEY (classid) REFERENCES dmhr.classes(calssid),
CONSTRAINT study_pk PRIMARY KEY (studentid, classid)
);
```

（3）附加说明。

① 表至少要包含一个字段，在一个表中，各字段名不能相同。一张表中最多可以包含 2048 个字段。

② 当字段类型为 DATE 类型时，指定默认值时，格式如 DEFAULT DATE '2005-13-26'，会对数据进行有效性检查。

③ 如果字段未指明 NOT NULL，也未指明<DEFAULT 子句>，则隐含为 DEFAULT NULL。

④ 如果完整性约束只涉及当前正在定义的列，则既可定义成列级完整性约束，也可以定义成表级完整性约束；如果完整性约束涉及该表的多个列，则只能在语句的后面定义成表级完整性约束。定义与该表有关的列级或表级完整性约束时，可以用 CONSTRAINT<约束名>子句对约束命名，系统中相同模式下的约束名不得重复。如果不指定约束名，系统将为此约束自动命名。经定义后的完整性约束被存入系统的数据字典中，用户操作数据库时，由 DBMS 自动检查该操作是否违背这些完整性约束条件。

2）用管理工具创建数据库表

【例 4-16】在 DMHR 模式下创建 DEPT 表，表的字段要求见表 4-7。

表 4-7　DEPT 表

字段名	字段类型	主键	非空	唯一
DEPTID	NUMBER(2, 0)	是	是	是
DEPTNAME	VARCHAR(20)		是	是
DEPTLOC	VARCHAR(128)			

步骤 1：在图 4-14 中右击 DMHR 模式下的表，在弹出的快捷菜单中选择“新建表”选项，弹出“新建表”对话框，如图 4-15 所示。

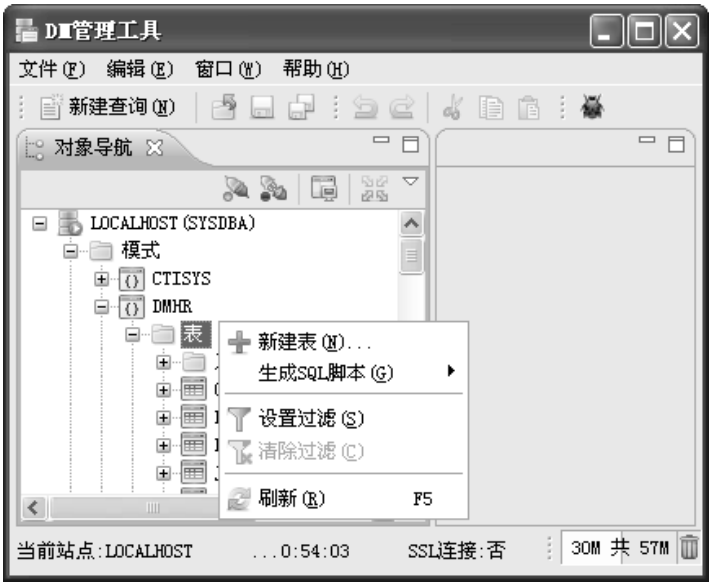

图 4-14　新建表

步骤 2：在图 4-15 中，进入常规参数页面，设置表名为“DEPT”，注释为“部门表”。

步骤 3：单击“+”按钮，增加一个字段，选中主键，列名为 DEPTID，数据类型选择 NUMBER，默认非空，精度为 2，标度为 0。

步骤 4：单击“+”按钮，增加一个字段，列名为 DEPTNAME，数据类型选择 VARCHAR，选中非空，精度为 20，标度为 0。在列属性中，值唯一选择“是”。

步骤 5：单击“+”按钮，增加一个字段，列名为 DEPTLOC，数据类型为 VARCHAR，精度为 128，标度为 0。

步骤 6：单击“确定”按钮，创建 DEPT 表成功。

图 4-15　设置表字段参数

【例 4-17】在 DMHR 模式下创建 EMP 表，表的字段要求见表 4-8 和表 4-9。

表 4-8　EMP 表

字段名	字段类型	主键	非空	唯一
EMPID	VARCHAR(5)	是	是	是
IDNUMBER	VARCHAR(18)		是	是
EMPNAME	VARCHAR(20)		是	
SEX	VARCHAR(2)			
DEPTID	NUMBER(2, 0)			

表 4-9　EMP 表的外键设置

字段名	外键	参考
DEPTID	是	DEPT(DEPTID)

步骤 1：在图 4-14 中右击 DMHR 模式下的表，在弹出的快捷菜单中选择“新建表”选项，弹出“新建表”对话框，如图 4-16 所示。

图 4-16　新建表的字段

步骤 2：在图 4-16 中，设置表名为“EMP”，注释为“员工表”。

步骤 3：单击“+”按钮，增加一个字段，选中主键，列名为 EMPID，数据类型选择 VARCHAR，默认非空，精度为 5，标度为 0。

步骤 4：单击“+”按钮，增加一个字段，列名为 IDNUMBER，数据类型选择 VARCHAR，选中非空，精度为 18，标度为 0。在列属性中，值唯一选择“是”。

步骤 5：单击“+”按钮，增加一个字段，列名为 EMPNAME，数据类型选择 VARCHAR，选中非空，精度为 20，标度为 0。

步骤 6：单击“+”按钮，增加一个字段，列名为 SEX，数据类型选择 VARCHAR，精度为 2，标度为 0。

步骤 7：单击“+”按钮，增加一个字段，列名为 DEPTID，数据类型选择 NUMBER，精度为 2，标度为 0。

步骤 8：进入选择约束参数页面，如图 4-17 所示，单击“添加”按钮，弹出“新建约束”对话框，如图 4-18 所示。

步骤 9：在图 4-18 中选中“外键约束”单选按钮，并单击“确定”按钮，弹出“配置外键约束”对话框，如图 4-19 所示。

图 4-17　设置约束

图 4-18　“新建约束”对话框

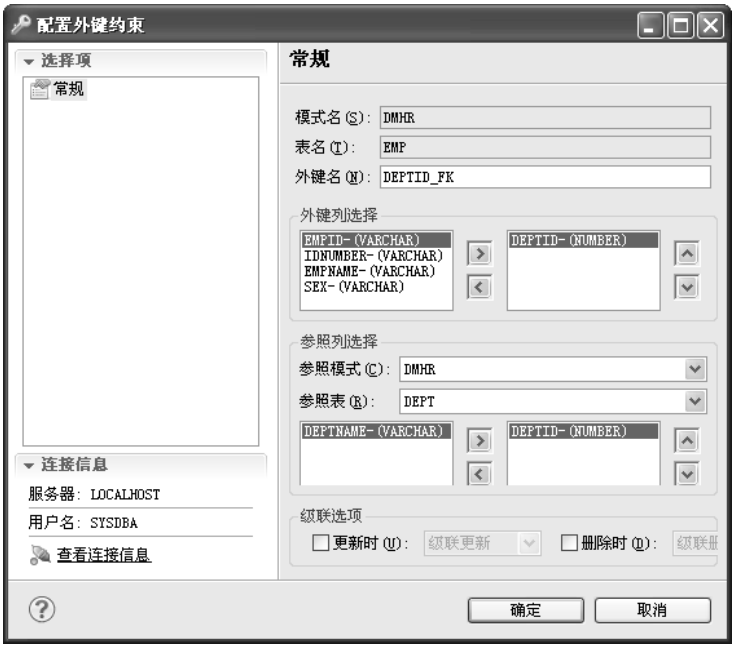

图 4-19　配置外键约束

步骤 10：在图 4-19 中输入外键名“DEPTID_FK”。

步骤 11：在“外键列选择”列表框中选择“DEPTID-(NUMBER)”字段，单击“>”

按钮。

步骤 12：在参照列中选择 DMHR 模式，选择 DEPT 为参照表，选择 DEPTID 为参照字段，单击“>”按钮，使 DEPTID 字段成为被参照字段。

步骤 13：单击“确定”按钮，完成外键配置，回到“新建表”对话框，如图 4-20 所示。

图 4-20　约束概况

步骤 14：在图 4-20 中单击“确定”按钮，完成 EMP 表创建过程。

2. 创建高性能数据库表

1）创建分区表

为了提高数据库的读、写性能，DM 数据库提供了分区表、HFS 表、LIST 表类型的数据表。随书光盘提供的 DM 数据库版本不提供分区表、HFS 表等功能，建议读者到达梦数据库有限公司官方网站（见附录 B）下载试用版。

（1）分区表概述。

① 应用背景：在大型的企业应用或企业级的数据库应用中，要处理的数据量通常达到 TB 级，对这样的大型表执行全表扫描或者 DML 操作时，效率是非常低的。

为了提高数据库在大数据量读写操作和查询时的效率，达梦数据库 DM 7 提供了对表和索引进行分区的技术，把表和索引等数据库对象中的数据分割成小的单位，分别存放在一个个单独的段中，用户对表的访问转化为对较小段的访问，以改善大型应用系统的性能。

例如，通信公司将用户通话记录保存在一张表中，一年这个表产生 40GB 的数据。假设要对用户的通话信息按照季度进行统计，那么这样的统计需要在全表范围内进行。如果对表按季度进行水平分区，那么每个分区的大小平均为 10GB 左右，这样在进行统计时，只需在 10GB 的范围内进行即可。由于 DM 7 划分的分区是相互独立且可以存储于不同

的存储介质上的，完全可满足企业高可用性、均衡 IO、降低维护成本、提高查询性能的要求。

DM 7 采用子表方式创建分区表，原表作为分区主表，而每一个分区以一个子表实体存在，即每一个分区都是一个完整的表，一般命名为“主表名_分区名”。在 DM 7 分区表中，主表本身不存储数据，所有数据只存储在子表中，从而实现不同分区的完全独立性。

分区操作对现存的应用和运行在分区表上的标准 DML 语句来说是透明的。但是，可以通过在 DML 中使用分区子表名称来对应用进行编程，使其充分利用分区的优点。

② 垂直分区表：垂直分区就是将表按列分拆为多个分区，每个分区子表包含较少的列，子表上的列是主表上列的子集。如果查询表上的某些列，并且这些列都在一个分区中，那么只需扫描这个分区即可，可以减少 I/O 量。

③ 水平分区表：对于水平分区，子表和主表具有相同的逻辑结构，即分区子表与分区主表有相同的列定义和约束定义。

范围分区：对表中的某些列上值的范围进行分区，根据某个值的范围，决定将该数据存储在哪个分区上。

哈希分区：通过指定分区编号来均匀分布数据的一种分区类型，通过在 I/O 设备上进行散列分区，使得这些分区大小基本一致。

列表分区：通过指定表中的某些列的离散值集，来确定应当存储在一起的数据。例如，可以对表上的 status 字段的值为（'A'，'H'，'O'）的放在一个分区，值为（'B'，'I'，'P'）的放在另一个分区，以此类推。

组合分区：按上述三种分区方式进行任意组合，将表进行多次分区，称为组合分区表。

（2）分区表语法格式：创建分区表的语法格式同创建数据库表的语法格式，重点在于<PARTITION 子句>。

```
<PARTITION 子句>::=PARTITION BY <PARTITION 项>
<PARTITION 项>::=RANGE (<列名>{,<列名>}) (<范围分区项>)|HASH (<列名>{,<列名>}) (<HASH 分区项>)|LIST(<列名>) (<LIST 分区项>)
<范围分区项>::=PARTITION <分区名> VALUES [EQU OR] LESS THAN (<表达式|MAXVALUE>{,<表达式|MAXVALUE>}) [<STORAGE 子句>][<子分区描述项>]
<HASH 分区项>::=PARTITION <分区名> [<STORAGE 子句>][<子分区描述项>]
<LIST 分区项>::=PARTITION <分区名> VALUES (DEFAULT|<<表达式>,{<表达式>}>) [<STORAGE 子句>][<子分区描述项>]
```

（3）用 SQL 命令创建分区表。

【例 4-18】创建范围分区表。在 DMHR 模式下创建一个范围分区表 CALLINFO，用来记录用户的 2016 年的电话通信信息，包括主叫号码、被叫号码、通话时间和时长，并且根据季度进行分区。

① 创建范围分区表 CALLINFO：

```
SQL> CREATE TABLE DMHR.callinfo
(
```

```
caller      CHAR(15),
callee      CHAR(15),
time        DATETIME,
duration    INT
)
PARTITION BY RANGE(time)
(
PARTITION p1 VALUES LESS THAN ('2016-04-01'),
PARTITION p2 VALUES LESS THAN ('2016-07-01'),
PARTITION p3 VALUES LESS THAN ('2016-10-01'),
PARTITION p4 VALUES LESS THAN ('2017-01-01')
);
警告: 范围分区未包含 MAXVALUE,可能无法定位到分区
```

该 SQL 命令的前半部分是创建一般表的格式，创建了 CALLINFO 表，该表包含 CALLER、CALLEE、TIME、DURATION 四个字段，后半部分是创建分区表的语法，根据 TIME 字段值来进行分区，P1 分区的 TIME 值小于'2016-04-01'，P2 分区的 TIME 值在'2016-04-01'与'2016-07-01'之间，P3 分区的 TIME 值在'2016-07-01'与'2016-10-01'之间，P4 分区的 TIME 值在'2016-10-01'与'2017-01-01'之间。命令执行后出现警告，原因是 2017 年以后的数据无法保存在该表中。

② 增加三条数据：

```
SQL> INSERT INTO DMHR.callinfo(caller, callee, time, duration) VALUES('13212345678', '13301234567', '2016-03-21 08:20:12', 12);
SQL> INSERT INTO DMHR.callinfo(caller, callee, time, duration) VALUES('13211223344', '13866554433', '2016-09-05 15:02:08', 58);
SQL> INSERT INTO DMHR.callinfo(caller, callee, time, duration) VALUES('15899887766', '13866554433', '2016-12-31 20:10:32', 135);
```

上述三条数据增加成功，试图增加一条超过时间超过 2017 年 1 月 1 日的数据。

```
SQL> INSERT INTO DMHR.callinfo(caller, callee, time, duration) VALUES('15899887766', '13866554433', '2017-03-06 20:10:32', 135);
 [-2731]:没有找到合适的分区.
```

第四条数据没有插入成功的原因是'2017-03-06 20:10:32'没有适合的分区区间，将不能保存。

③ 直接查询数据：

```
SQL> SELECT * FROM DMHR.callinfo;
行号        CALLER          CALLEE          TIME            DURATION
---------- --------------- --------------- ------------------------- ----------------------------------------
```

```
1          13212345678     13301234567     2016-03-21 08:20:12.000000   12
2          13211223344     13866554433     2016-09-05 15:02:08.000000   58
3          15899887766     13866554433     2016-12-31 20:10:32.000000   135
```

④ 按分区查询数据，查询第三季度的通话：

```
SQL> SELECT * FROM DMHR.callinfo partition(p3);
行号        CALLER          CALLEE          TIME                       DURATION
---------- --------------- --------------- -------------------------- -----------------------------------------
1          13211223344     13866554433     2016-09-05 15:02:08.000000   58
```

这个例子说句，查询第三季的通话，实际上是查询 CALLINFO 分区表的第 P3 分区。

（4）用管理工具创建分区表。

【例 4-19】创建范围分区表。在 DMHR 模式下创建一个范围分区表 CALLINFO2，用来记录用户的 2016 年的电话通信信息，包括主叫号码、被叫号码、通话时间和时长，并且根据季度进行分区。

步骤 1：在图 4-21 中，右击 DMHR 模式下的表，在弹出的快捷菜单中选择“新建表”选项，弹出“新建表”对话框，如图 4-22 所示。

图 4-21　新建表

步骤 2：在图 4-22 中，设置表名为“CALLINFO2”。增加 CALLER 字段，类型为 CHAR，长度为 15；增加 CALLEE 字段，类型为 CHAR，长度为 15；增加 TIME 字段，类型为 DATATIME，长度默认；增加 DURATION 字段，类型为 INT，长度为默认。

步骤 3：进入分区参数页面，选中“创建为分区表”复选框，分区类型选择“范围分区”，分区列选择 TIME，增加四个分区，并设置表达式和边界值方式，如图 4-23 所示。

图 4-22　增加表字段

图 4-23　设置范围分区参数

步骤 4：在图 4-23 中，单击“确定”按钮，完成 CALLINFO2 分区表创建过程。

2）创建 HFS 表

（1）HFS 表概述。

HFS 表是建立在 HTS 表空间上的。在创建一个 HFS 表之后，数据库会在指定的 HTS 表空间目录下创建一系列的目录及文件，文件系统结构如图 4-24 所示。

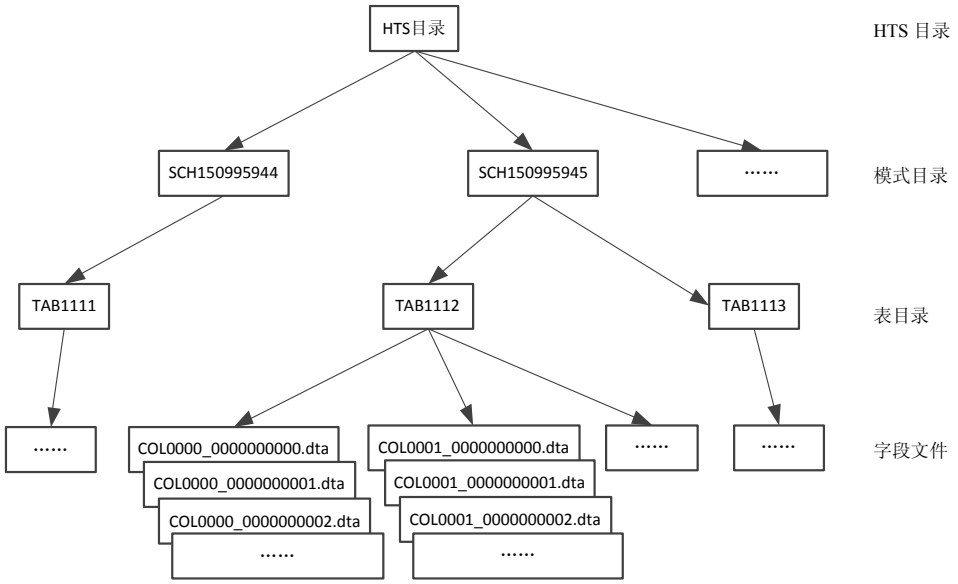

图 4-24　HFS 结构示意图

从上图可以看出，在 HTS 目录下成功创建 HFS 表，系统内部需要经过以下步骤。

首先，在 HTS 目录下创建这个表对应的模式目录，目录名为“SCH+长度为 9 的 ID”组成的字符串，如 SCH150995944。创建时如果这个目录已经存在，则无需重新创建。

其次，在模式目录下创建对应的表目录。表目录也是同样的道理，表目录名为“TAB+长度为 4 的 ID”组成的字符串，如 TAB1112。表目录中存放的是这个表中所有的文件。

再次，在新创建表时，每一个字段对应一个以.dta 为后缀的文件，文件大小可以在建表时指定，默认为 64MB，文件名为“COL+长度为 4 的列号+_+长度为 10 的文件号”。例如，在图 4-24 中，0000 表示第 1 列，0001 表示第 2 列……；0000000000 表示第 1 个文件，0000000001 表示第二个文件……最初一个列只有一个文件即可，但随着数据量不断增长，一个文件已放不下之后，系统会自动创建新的文件来存储不断增长的数据。

HFS 表主要针对海量数据的扫描分析，并不适用于常规的联机事务处理。

（2）语法格式。

CREATE HUGE TABLE <表名定义> <表结构定义>[<PARTITION 子句>] [<STORAGE 子句 1>][<压缩子句] [<DISTRIBUTE 子句>][<日志属性>];

其中，各子句说明如下：

<STORAGE 子句 1>::=STORAGE([SECTION (<区大小>)][FILESIZE (<文件大小>)][STAT NONE] ON <HTS 表空间名>)

<压缩子句> ::= COMPRESS [LEVEL <压缩级别>] [FOR 'QUERY [LOW | HIGH]'] |COMPRESS [LEVEL <压缩级别>] [FOR 'QUERY [LOW | HIGH]'] (<列名> [LEVEL <压缩级别>] [FOR 'QUERY [LOW | HIGH]'] {,<列名> [LEVEL <压缩级别>] [FOR 'QUERY [LOW| HIGH]'] }) |COMPRESS [LEVEL <压缩级别>] [FOR 'QUERY [LOW | HIGH]'] EXCEPT (<列名>{,<列名>})

<DISTRIBUTE 子句> ::= DISTRIBUTED[<RANDOMLY>|<FULLY>] |DISTRIBUTED BY

[<HASH>](<列名> {,<列名>})|DISTRIBUTED BY RANGE (<列名> {,<列名>})(<范围分布项> {,<范围分布项>}) |DISTRIBUTED BY LIST (DEFAULT|<<列名> {,<列名>}>)(<列表分布项> {,<列表分布项>})

<范围分布项> ::= VALUES LESS THAN (<表达式>{,<表达式>}) ON <实例名> |VALUES EQU OR LESS THAN (<表达式>{,<表达式>}) ON <实例名>

<列表分布项> ::= VALUES (<表达式>{,<表达式>}) ON <实例名>

<日志属性>::=LOG NONE|LOG LAST|LOG ALL

参数说明见表 4-10。

表 4-10　参数说明

参　数	说　明
<表名>	指明被创建的 HFS 表名，HFS 表名最大长度为 124 字节
<区大小>	指一个区的数据行数。区的大小必须是 2 的多少次方，如果不是则向上对齐。取值范围：1024 行～1024×1024 行。不指定时默认值为 65536 行
[STAT NONE]	指定不记录区统计信息，即在修改时不做数据的统计。不指定情况下，默认为做统计信息
<HTS 表空间名>	指要创建的 HFS 表所属的 HTS 表空间。 不指定则存储于默认系统 HFS 表空间 HMAIN 中
<文件大小>	指创建 HFS 表时指定的单个文件的大小。取值为 16MB～1024×1024MB。文件大小必须是 2 的多少次方，如果不是则向上对齐。不指定时默认为 64MB
<压缩级别>	为特定列指定压缩级别，有效值为 0～9。其中，值越小表示压缩速率越快，值越大表示压缩比越高。0，1 为快捷使用，默认值为 0，表示最大速率压缩，等价于 LEVEL2；设置成 1 时表示最高压缩比，即等价于 LEVEL 9
<日志属性>	支持通过做日志来保证数据的完整性。完整性保证策略主要是通过数据的镜像来实现的，镜像的不同程度可以实现不同程度的完整性恢复。有以下三种选择： 1）LOG NONE：不做镜像。相当于不做数据一致性的保证，如果出错只能手动通过系统函数 SF_REPAIR_HFS_TABLE(模式名,表名)来修复表数据。 2）LOG LAST：做部分镜像。但是在任何时候都只对当前操作的区做镜像，如果当前区的操作完成了，这个镜像也就失效了，并且可能会被下一个被操作区覆盖，这样做的好处是镜像文件不会太大，同时可以保证数据是完整的。但有可能遇到的问题是，在一次操作很多的情况下，有可能一部分数据已经完成，另一部分数据还没有来得及做。 3）LOG ALL：全部做镜像。在操作过程中，所有被修改的区都会被记录下来，当一次操作修改的数据过多时，镜像文件有可能会很大，但能够保证操作完整性。默认选择为 LOG LAST

（3）用 SQL 命令创建 HFS 表。

【例 4-20】以用户 SYSDBA 登录，在 HTS1 表空间创建 HFS 表 ORDERS，包含 ORDERID、TOTALPRICE、ORDERDATE、ORDERCOMMENT 等字段，表的区大小为 65536 行，文件大小为 64MB，指定所在的表空间为 HTS1，做完整镜像。ORDERCOMMENT 字段不做统计信息，其他字段都做统计信息，ORDERCOMMENT 字段按照最大压缩比压缩。

```
SQL> CREATE HUGE TABLE dmhr.orders
(
orderid    INT,
totalprice FLOAT,
orderdate    DATE,
ordercomment    VARCHAR(79) STORAGE(STAT NONE)
)
STORAGE(SECTION(65536), FILESIZE(64), ON HTS1)
COMPRESS LEVEL 1(ORDERCOMMENT)
LOG ALL;
```

该 SQL 命令使用 HUGE 关键词来创建 HFS 表。前半部分是创建一般表的格式，创建了 ORDERS 表，该表包含 ORDERID、TOTALPRICE、ORDERDATE、ORDERCOMMENT 四个字段，并且 OERDERCOMMENT 字段不做统计信息。后半部分是设置存储格式，区大小为 65536 行，文件大小为 64MB，存储在 HTS1 表空间中，做完整镜像；ORDERCOMMENT 字段按最大压缩比压缩。

（4）附加说明。

① HFS 表操作时封锁粒度较大，且不支持多版本并发控制。

② HFS 表的插入、删除与更新操作处理都不能进行回滚。

③ 创建 HFS 表时仅支持定义 NULL、NOT NULL、UNIQUE 约束以及 PRIMARY KEY，后两种约束也可以通过 ALTER TABLE 的方式添加，但这两种约束不检查唯一性。

④ HFS 允许建立二级索引，其中 UNIQUE 索引不检查唯一性。

⑤ HFS 表不支持事务，没有事务的特性。

⑥ 不支持 SPACE LIMIT（空间限制）。

⑦ 不支持 IDENTITY 自增列。

⑧ 不支持大字段列。

（5）用管理工具创建 HFS 表。

【例 4-21】以用户 SYSDBA 登录，在 HTS1 表空间创建 HFS 表 ORDERS2，包含 ORDERID、TOTALPRICE、ORDERDATE、ORDERCOMMENT 等字段，表的区大小为 65536 行，文件大小为 64MB，指定所在的表空间为 HTS1，做完整镜像。ORDERCOMMENT 字段不做统计信息，其他字段都做统计信息，ORDERCOMMENT 字段按照最大压缩比压缩。

步骤 1：在图 4-25 中右击 DMHR 模式下的表，在弹出的快捷菜单中选择“新建表”选项，弹出“新建表”对话框，如图 4-26 所示。

步骤 2：在图 4-26 中，设置表名为 ORDERS2，增加 ORDERID、TOTALPRICE、ORDERDATE、ORDERCOMMENT 等字段，并设置相应字段类型。

步骤 3：进入选项页面，如图 4-27 所示，表类型选择“HUGE 表”，日志选项设置为“记录所有”。

图 4-25　新建表

图 4-26　增加字段

图 4-27　设置选项参数

步骤 4：进入存储页面，如图 4-28 所示，表空间设置为 HTS1，HUGE 表区大小设置为 65536，HUGE 表文件大小设置为 64，非数据统计列选择“ORDERCOMMENT”，压缩方式为“指定压缩列”，选择压缩列为“ORDERCOMMENT”，压缩级别设置为“1”。

图 4-28　设置存储参数

步骤 5：在图 4-28 中，单击“确定”按钮，完成 HUGE 表创建过程。

3）创建 LIST 表

（1）LIST 表概述。

普通表都是以 B 树形式存放的，ROWID 都是逻辑的 ROWID，即从 1 一直增长下去。在并发情况下，每次插入过程中都需要逻辑生成 ROWID，这样影响了插入数据的效率；对于每一条数据都需要存储 ROWID 值，也会花费较大的存储空间。LIST 表就是基于上述两个理由而提出的。

简单地说，LIST 表是指采用了物理 ROWID 形式的表，即使用文件号、页号和页内偏移而得到 ROWID 值，这样就不需要存储 ROWID 值，可以节省空间。逻辑 ROWID 在插入或修改过程中，为了确保 ROWID 的唯一性，需要依次累加而得到值，这样就影响了效率，而 LIST 表只需根据自己的文件号、页号和页内偏移就可以得到 ROWID，提高了效率。普通表都是以 B 树形式而存储在物理磁盘上的，而 LIST 表则采用一种“扁平 B 树”方式存储，结构如图 4-29 所示。

采用了物理 ROWID 形式的 LIST 表，DM 服务器内部对聚集索引进行了调整，没有采用传统 B 树结构，取而代之的是“扁平 B 树”，数据页都是通过链表形式存储的。为支持并发插入，扁平 B 树可以支持最多 128 个数据页链表（最多 64 个并发分支和最多 64 个非并发分支），在 B 树的控制页中记录了所有链表的首、尾页地址。对于并发分支，则不同

用户会选择不同的分支来进行插入，如果存在多个用户选择了同一条分支的情况，才需要等待其他用户插入结束并释放锁之后再进行插入。在并发情况下，不同用户可以在不同的链表上进行插入，效率得到了较大提升。

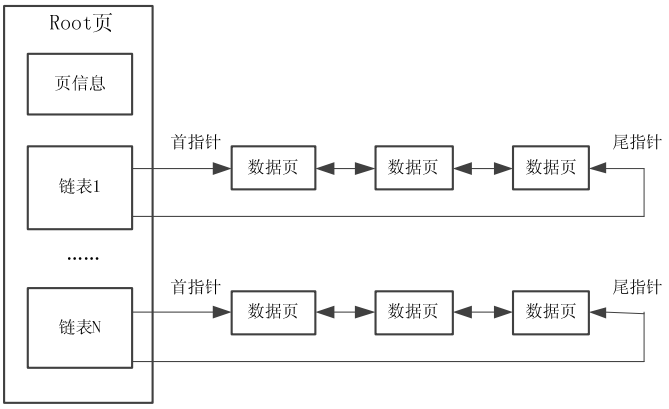

图 4-29　LIST 表存储方式

（2）LIST 表创建方法。

LIST 表的创建有两种方式，一种是在配置文件 dm.ini 中设置参数，另一种是在建表 SQL 语句中显式指定 LIST 表选项。

① INI 参数方式：用户可以在配置文件 dm.ini 中添加 LIST_TABLE 参数。

如果 LIST_TABLE=1，则在未显式指定表是否为 LIST 表或非 LIST 表时，默认情况下创建的表为 LIST 表。

如果 LIST_TABLE=0，则在未显式指定表是否为 LIST 表或非 LIST 表时，默认情况下创建的表为普通表形式。

② SQL 语句显示指定：不管参数 LIST_TABLE 设置为何值，创建表时可以在 STORAGE 选项中指定需要创建的表形式，与 LIST 表创建形式相关的关键字有三个，分别是 NOBRANCH、BRANCH、CLUSTERBTR。

NOBRANCH：创建的表为 LIST 表，并发分支个数为 0，非并发分支个数为 1。

BRANCH(n，m)：创建的表为 LIST 表，并发分支个数为 n，非并发个数为 m。

CLUSTERBTR：创建的表为非 LIST 表，即普通 B 树表。

（3）用 SQL 命令创建 LIST 表。

【例 4-22】以用户 SYSDBA 在 DMHR 模式下创建 LISTSTUDENT 表，包含 ID、NAME、BIRTHDAY 等字段，并发分支数为 5，非并发分支数为 5。

```
SQL> CREATE TABLE dmhr.liststudent
(
id   INT,
name   VARCHAR(30) ,
birthday   DATE
)
STORAGE(BRANCH(5,5));
```

该 SQL 命令前半部分是创建普通表的格式，LISTSTUDENT 表包含了 ID、NAME、BIRTHDAY 三个字段；后半部分设置 LIST 表的分支数，5 个并发分支，5 个非并发分支。

（4）用管理工具创建 LIST 表。

【例 4-23】以用户 SYSDBA 在 DMHR 模式下创建 LISTSTUDENT2 表，包含 ID、NAME、BIRTHDAY 等字段，并发分支数为 5，非并发分支数为 5。

步骤 1：在图 4-30 中右击 DMHR 模式下的表，在弹出的快捷菜单中选择“新建表”选项，弹出“新建表”对话框，如图 4-31 所示。

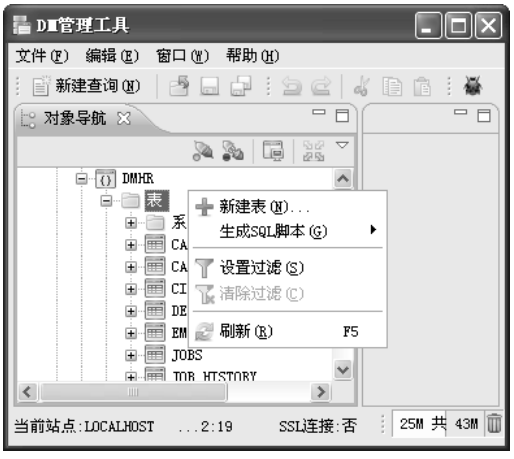

图 4-30　新建表

步骤 2：在图 4-31 中，设置表名为“LISTSTUDENT2”，增加 ID、NAME、BIRTHDAY 等字段，并设置相应的字段类型。

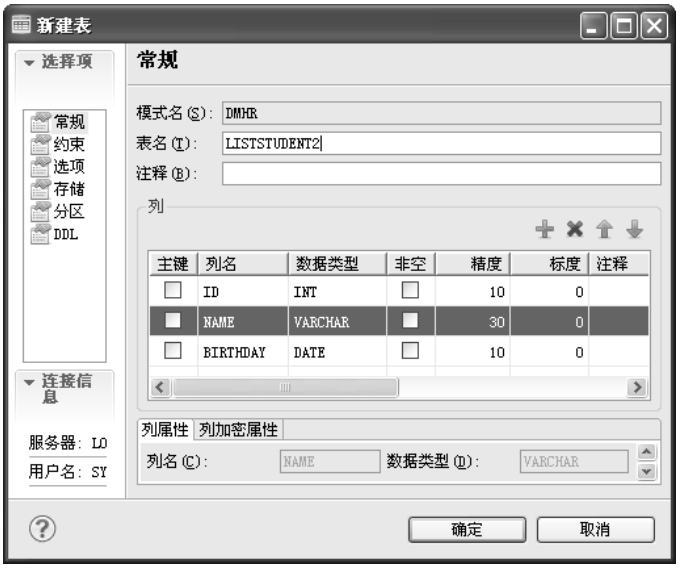

图 4-31　增加字段

步骤 3：进入存储页面，如图 4-32 所示，设置 LIST 表选项为双分支，分支数为“5, 5”。

步骤 4：单击“确定”按钮，完成 LIST 表的创建过程。

图 4-32　设置存储参数

3. 修改数据库表

为了满足用户在建立应用系统的过程中需要调整数据库结构的要求，DM 系统提供了数据库表修改语句，对表的结构进行全面的修改，包括修改表名、字段名、增加字段、删除字段、修改字段类型、增加表级约束、删除表级约束、设置字段默认值、设置触发器状态等一系列修改。

1）用 SQL 命令修改数据库表

（1）语法格式。

修改数据库表的 SQL 命令格式如下：

```
ALTER TABLE [<模式名>.]<表名> <修改表定义子句>
```

其中，<修改表定义子句>简化格式如下：

```
MODIFY <字段定义>|
ADD [COLUMN] <字段定义>|
DROP [COLUMN] <字段名> [RESTRICT|CASCADE] |
ADD [CONSTRAINT [<约束名>]] <表级约束定义> [<CHECK 选项>]|
DROP CONSTRAINT <约束名> [RESTRICT | CASCADE]
```

（2）应用举例。

【例 4-24】修改字段类型长度。以 SYSDBA 用户登录，将 PRODUCTION 模式的 PRODUCT_VIEW 表的 NAME 字段的数据类型改为 VARCHAR(30)，并指定该列为 NOT NULL，且默认值为'刘青'。

```
SQL> ALTER TABLE production.product_review MODIFY name VARCHAR(30) NOT NULL DEFAULT '刘青' ;
```

【例 4-25】增加普通字段。以 SYSDBA 用户登录，给 RESOURCES 模式的 EMPLOYEE_ADDRESS 表增加 USEDATE 字段，字段类型为 DATE。

```
SQL> ALTER TABLE resources.employee_address ADD usedate DATE;
```

【例 4-26】增加唯一值字段。以 SYSDBA 用户登录，给 RESOURCES 模式的 EMPLOYEE_ADDRESS 表增加 NUM 字段，字段类型为 INT，且为唯一值字段。

```
SQL> ALTER TABLE resources.employee_address ADD num INT UNIQUE;
```

【例 4-27】增加主键字段。以 SYSDBA 用户登录，给 RESOURCES 模式的 EMPLOYEE_ADDRESS 表增加 ID 字段，字段类型为 INT，且为主键字段。

```
SQL> ALTER TABLE resources.employee_address ADD id INT PRIMARY KEY;
ALTER TABLE resources.employee_address ADD id INT PRIMARY KEY;
[-6609]:违反列[ID]非空约束.
```

该命令执行失败是由于 EMPLOYEE_ADDRESS 已经有数据记录，增加主键字段时，由于主键字段为空，违反了实体完整性约束，所以增加主键字段不成功。

【例 4-28】增加 CHECK 约束。以 SYSDBA 用户登录，给 RESOURCES 模式的 EMPLOYEE_ADDRESS 表增加 CHECK 约束，名称为 NUM_CHECK，要求 NUM 字段的值大于 1000。

```
SQL> ALTER TABLE resources.employee_address ADD CONSTRAINT num_check CHECK (num>1000);
```

【例 4-29】删除约束。以 SYSDBA 用户登录，删除 RESOURCES 模式的 EMPLOYEE_ADDRESS 表的 NUM_CHECK 约束。

```
SQL> ALTER TABLE resources.employee_address DROP CONSTRAINT num_check;
```

【例 4-30】删除字段。以 SYSDBA 用户登录，删除 RESOURCES 模式的 EMPLOYEE_ADDRESS 表的 NUM 字段。

```
SQL> ALTER TABLE resources.employee_address DROP num CASCADE;
```

（3）附加说明。

① 使用 MODIFY COLUMN 时，不能更改聚集索引的列或者引用约束中引用和被引用的列。

② 使用 MODIFY COLUMN 时，一般不能更改用于 CHECK 约束的列。只有当该 CHECK 列都为字符串，且新列的长度大于旧列长度，或都为整型时，新列的类型才能够完全覆盖旧列的类型（如 char（1）到 char（20），tinyint 到 int）时才能修改。

③ 使用 MODIFY COLUMN 子句不能在列上增加 CHECK 约束，能修改的约束只有列上的 NULL、NOT NULL 约束；如果某列现有的值均非空，则允许添加 NOT NULL；属于聚集索引包含的列不能被修改；自增列不允许被修改。

④ 使用 MODIFY COLUMN 修改可更改列的数据类型时，若该表中无元组，则可任意修改其数据类型、长度、精度或量度；若表中有元组，则系统会尝试修改其数据类型、长度、精度或量度，如果修改不成功，则会报错返回。无论表中有无元组，多媒体数据类

型和非多媒体数据类型都不能相互转换。

⑤ 修改有默认值的列的数据类型时，原数据类型与新数据类型必须是可以转换的，否则即使数据类型修改成功，在进行插入等其他操作时，仍会出现数据类型转换错误。

⑥ 使用 ADD COLUMN 时，新增列名之间、新增列名与该基表中的其他列名之间均不能重复。若新增列跟有默认值，则已存在的行的新增列值是其默认值。添加新列对于任何涉及表的约束定义没有影响，对于涉及表的视图定义会自动增加。例如，如果用“*”为一个表创建一个视图，那么后加入的新列会自动地加入该视图。

⑦ 使用 ADD COLUMN 时，还有以下限制条件。

a. 列定义中如果带有列约束，只能是对该新增列的约束，但不支持引用约束；列级约束可以带有约束名，系统中同一模式下的约束名不得重复，如果不带约束名，系统自动为此约束命名。

b. 如果表上没有元组，列可以指定为 NOT NULL；如果表中有元组，对于已有列可以指定同时有 DEFAULT 和 NOT NULL，新增列不能指定 NOT NULL。

c. 该列可指定为 CHECK。

d. 该列可指定为 FOREIGN KEY。

e. 允许向空数据的表中添加自增列。

⑧ ADD CONSTRAINT 子句用于添加表级约束。表级约束包括：主键约束（PRIMARY KEY）、唯一性约束（UNIQUE）、引用约束（REFERENCES）、检查约束（CHECK）。添加表级约束时可以带有约束名，系统中同一模式下的约束名不得重复，如果不带约束名，系统自动为此约束命名；用 ADD CONSTRAINT 子句添加约束时，对于该基表上现有的全部元组要进行约束违规验证。

a. 添加一个主键约束时，要求将成为关键字的字段上无重复值且值非空，并且表上没有定义主关键字。

b. 添加一个 UNIQUE 约束时，要求将成为唯一性约束的字段上不存在重复值，但允许有空值。

c. 添加一个 REFERENCES 约束时，要求将成为引用约束的字段上的值满足该引用约束。

d. 添加一个 CHECK 约束或外键时，要求该基表中全部的元组满足该约束。

⑨ 用 DROP COLUMN 子句删除一列有两种方式：RESTRICT 和 CASCADE。RESTRICT 方式为默认选项，确保只有不被其他对象引用的列才被删除。无论哪种方式，表中的唯一列不能被删除。RESTRICT 方式下，下列类型的列不能被删除：被引用列、建有视图的列、有 CHECK 约束的列。删除列的同时将删除该列上的约束。CASCADE 方式下，将删除这一列上的引用信息和被引用信息、引用该列的视图、索引和约束；系统允许直接删除 PK 列。但被删除列为 CLUSTER PRIMARY KEY 类型时除外，此时不允许删除。

⑩ DROP CONSTRAINT 子句用于删除表级约束，表级约束包括：主键约束（PRIMARY KEY）、唯一性约束（UNIQUE）、引用约束（REFERENCES）、检查约束（CHECK）。用 DROP CONSTRAINT 子句删除约束时，同样有 RESTRICT 和 CASCADE 两种方式。当删除主键或唯一性约束时，系统自动创建的索引也将一起删除。如果打算删除一个主键约束

或一个唯一性约束而它有外部约束，除非指定 CASCADE 选项，否则将不允许删除。也就是说，指定 CASCADE 时，删除的不仅仅是用户命名的约束，还有任何引用它的外部约束。

⑪ 各个子句中都可以含有单个或多个列定义（约束定义），单个列定义（约束定义）和多个列定义（约束定义）应该在它们的定义外加一层括号，括号要配对。

⑫ 具有 DBA 权限的用户或该表的建表者才能执行此操作。

2）用管理工具修改数据库表

【例 4-31】删除字段。以 SYSDBA 用户登录，删除 RESOURCES 模式的 EMPLOYEE_ADDRESS 表中的 USEDATE 字段。

步骤 1：在图 4-33 中右击 USEDATE 字段，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-34 所示。

图 4-33　删除字段

图 4-34　确认删除字段

步骤 2：在图 4-34 中，单击“确定”按钮，确认删除 USEDATE 字段。

【例 4-32】增加字段。以 SYSDBA 用户登录，给 RESOURCES 模式的 EMPLOYEE_ADDRESS 表增加 USEDATE 字段，字段类型为 DATE。

步骤 1：右击 EMPLOYEE_ADDRESS 字段，在弹出的快捷菜单中选择“修改”选项，弹出“修改表”对话框，如图 4-35 所示。

图 4-35　增加 USEDATE 字段

步骤 2：在图 4-35 中单击“+”按钮，增加字段，字段名为 USEDATE，字段类型改为 DATE，默认值改为'2016-3-31'。

步骤 3：在图 4-35 中，单击“确定”按钮保存修改。

4. 删除数据库表

删除数据库表会导致该表的数据以及对该表的约束依赖被删除，因此业务工作中很少有删除数据表的操作，但是作为数据库管理员，掌握删除数据库表的方法是非常必要的。

1）用 SQL 命令删除数据库表

（1）语法格式。

删除数据库表的 SQL 命令如下：

```
DROP TABLE [<模式名>.]<表名> [RESTRICT|CASCADE];
```

表删除有两种方式：RESTRICT/CASCADE 方式，其中 RESTRICT 为默认值。如果以 RESTRICT 方式删除该表，要求该表上已不存在任何视图以及参照完整性约束，否则 DM 返回错误信息，而不删除该表。如果以 CASCADE 方式删除该表，将删除表中唯一列上和主关键字上的参照完整性约束，当设置 dm.ini 中的参数 DROP_CASCADE_VIEW 值为 1 时，还可以删除所有建立在该表上的视图。

（2）应用举例。

在 DMHR 模式下，EMP 表（员工表）参照了 DEPT 表（部门表），在这里 DEPT 称之为被参照表，EMP 表称之为参照表。

【例 4-33】删除被参照表。删除 DMHR 模式下的 DEPT（部门表）。

```
SQL> DROP TABLE dmhr.dept;
DROP TABLE dmhr.dept;
第 1 行附近出现错误[-2639]:试图删除被依赖对象[DEPT].
```

这项任务执行出现错误，原因是不能简单直接删除被参照表。

【例 4-34】删除参照表。删除 DMHR 模式下的 EMP 表（员工表）。

```
SQL> DROP TABLE dmhr.emp;
```

表删除后，与被参照表的参照完整性约束也一同被删除。

【例 4-35】在删除 EMP 表的基础上，再次删除 DEPT 表。

```
SQL> DROP TABLE dmhr.dept;
```

由于没有与 DEPT 关联的引用约束，所以该任务能够顺利执行。

（3）附加说明。

① 该表删除后，在该表上所建索引也同时被删除。

② 该表删除后，所有用户在该表上的权限也自动取消，以后系统中再创建的同名基表是与该表毫无关系的表。

2）用管理工具删除数据库表

【例 4-36】删除被参照表。删除 DMHR 模式下的 CLASSES 表和 STUDENTS 表。

步骤 1：右击 CLASSES 表，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框如图 4-36 所示，选中“级联删除”复选框，单击“确定”按钮，确认删除。

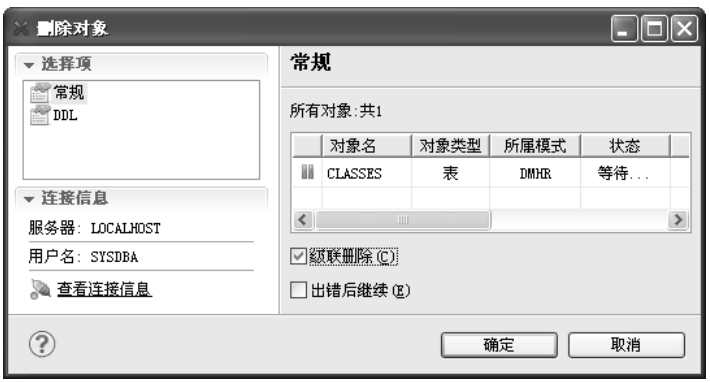

图 4-36　删除 CLASSES 表

步骤 2：右击 STUDENTS 表，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-37 所示，选中“级联删除”复选框，单击“确定”按钮，确认删除。

图 4-37　删除 STUDENTS 表

【例 4-37】删除参照表。删除 DMHR 模式下的 STUDY 表。

右击 STUDY 表，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-38 所示，单击“确定”按钮，确认删除。

图 4-38　删除 STUDY 表

4.3.2　管理外部表

外部表的定义存储在数据库中，外部表的数据存储在数据库外部的文件中，在控制文件的支持下，可以按照数据库外部表的定义从外部文件中读取数据。在数据库中只能从外部表读数据，不能向外部表写数据，也不能删除数据。DM 7 不提供外部表的修改语句，如果想更改外部表的表结构，可以通过重建外部表来实现。

1. 创建外部表

创建外部表的基础是建立数据文件和控制文件，可以通过 SQL 命令和管理工具两种方式来创建外部表。

1）用 SQL 命令创建外部表

（1）语法格式。

创建外部表的 SQL 命令格式如下：

```
CREATE EXTERNAL TABLE <表名> (<字段定义> {,<字段定义>}) FROM <控制文件路径>;
```

字段暂不支持多媒体类型，控制文件路径是一个包含控制文件路径和文件名的字符串。

控制文件的格式如下：

```
LOAD [DATA]
INFILE [LIST] <path_name>|<path_name_list>
INTO TABLE tablename
FIELDS <delimiter>
```

其中：

<path_name_list>——数据文件列表；

Tablename——表名；

<delimiter>——同一行中各个字段的分隔符。

数据文件中一行数据必须以回车结束，数据文件的字段值之间以<delimiter>分隔开。与数据库表相比，外部基表不能有大字段列，不能有任何约束条件，不能为临时表，不能建立分区，不能建立任何索引。

外部表支持查询 ROWID、USER 和 UID 伪列，不支持查询 TRXID 伪列。

（2）应用举例。

【例 4-38】指定操作系统的一个文本文件为数据文件，编写控制文件，创建外部表 EXT。

① 编辑数据文件（C:\SQL\DATA1.TXT），内容如下：

```
李建国|1973-10-10|62.5|
张泽涛|1977-06-21|76.8
```

② 编辑控制文件（C:\SQL\CTRL.TXT），内容如下：

```
LOAD DATA
INFILE 'C:\SQL\DATA1.TXT'
INTO TABLE EXT
FIELDS '|'
```

③ 创建外部表 EXT：

```
SQL> CONN DMHR/DMHR12345;
SQL> CREATE EXTERNAL TABLE EXT
(
tname VARCHAR(30),
tbirthday DATE,
tweight NUMBER(4,1)
)
FROM 'C:\SQL\CTRL.TXT';
```

该 SQL 命令使用了 EXTERNAL 关键字，前半部分与普通表的定义相同，EXT 表包含 TNAME、TBIRTHDAY、TWEIGHT 三个字段；后半部分指明控制文件的绝对路径。

④ 查询 EXT 表的数据：

```
SQL> SELECT * FROM EXT;
```

```
行号        TNAME      TBIRTHDAY       TWEIGHT
---------- ------ ---------- ----------------------------------------
1           李建国        1973-10-10          62.5
2           张泽涛        1977-06-21          76.8
```

在数据库中只能查询外部表的数据，查询方法与查询数据库表的方法相同。

⑤ 增、删、改 EXT 表的数据：

由于外部表、数据是只读的，在数据库系统内不能增加、修改和删除 EXT 表的数据，如果要对 EXT 表做增、删、改操作，只能编辑外部文件 C:\SQL\DATA1.TXT，对这个外部文件删除一条数据（李建国），修改一条数据（张泽涛的体重改为 66.8），增加一条数据（刘永胜），内容如下：

```
张泽涛|1977-06-21|66.8|
刘永胜|1980-12-15|73.6|
```

⑥ 查询增、删、改后的 EXT 表数据：

```
SQL> SELECT * FROM dmhr.ext;
行号        TNAME      TBIRTHDAY       TWEIGHT
---------- ------ ---------- ----------------------------------------
1           张泽涛        1977-06-21          66.8
2           刘永胜        1980-12-15          73.6
```

2）用管理工具创建外部表

【例 4-39】指定操作系统的一个文本文件作为数据文件，编写控制文件，创建外部表 EXT2。

步骤 1：在图 4-39 中右击 DMHR 模式下的外部表，在弹出的快捷菜单中选择“新建外部表”选项，弹出“新建外部表”对话框，如图 4-40 所示。

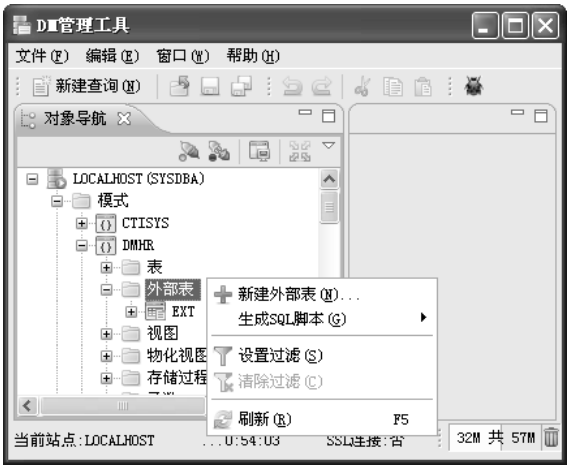

图 4-39　新建外部表

步骤 2：在图 4-40 中，设置表名为“EXT2”，控制文件路径为“C:\SQL\CTRL.txt”。

步骤 3：单击“+”按钮，新增表字段，列名为 XNAME，字段类型为 VARCHAR，精

度为 30，标度为 0。

步骤 4：单击“+”按钮，新增表字段，列名为 XBIRTHDAY，字段类型为 DATE，精度为 10，标度为 0。

步骤 5：单击“+”按钮，新增表字段，列名为 XWEIGHT，字段类型为 NUMBER，精度为 4，标度为 1。

图 4-40　设置外部表名和字段

步骤 6：单击“确定”按钮，外部表 EXT2 创建完成。

步骤 7：右击 EXT2 表，在弹出的快捷菜单中选择“浏览数据”选项，结果如图 4-41 所示，表明外部表创建成功。

图 4-41　查询外部表数据

2. 删除外部表

删除外部表与删除数据库表的方法相同。

1）用 SQL 命令删除外部表

【例 4-40】删除 DMHR 模式下的外部表 EXT。

```
SQL> DROP TABLE dmhr.ext;
```

2）用管理工具删除外部表

【例 4-41】删除 DMHR 模式下的外部表 EXT2。

步骤 1：在图 4-42 中，右击 DMHR 模式下的外部表 EXT2，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-43 所示。

图 4-42　删除外部表

图 4-43　确认删除外部表

步骤 2：在图 4-43 中，确认 EXT2 设定无误后，单击“确定”按钮，完成删除外部表的过程。

4.4　视图管理

视图包括普通视图和物化视图。普通视图是一个虚表，从视图中可以查阅数据但并不真正存储数据。物化视图是从一个或几个基表导出的表，同普通视图相比，它存储了导出表的真实数据。

通常把普通视图直接称为视图，在本书中，如果上下文没有特别说明，视图就是指普通视图。

4.4.1 创建视图

视图是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制，它简化了用户数据模型，提供了逻辑数据独立性，实现了数据共享和数据的安全保密。视图是数据库技术中一个十分重要的功能。从系统实现的角度讲，视图是从一个或几个基表（或视图）导出的表，但它是一个虚表，即数据字典中只存放视图的定义（由视图名和查询语句组成），而不存放对应的数据，这些数据仍存放在原来的基表中。当对一个视图进行查询时，视图将查询其对应的基表，并且将所查询的结果以视图所规定的格式和次序进行返回。因此，当基表中的数据发生变化时，从视图中查询出的数据也随之改变了。从用户的角度来讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据和变化。当用户所需的数据是一张表的部分列或部分行，或者数据分散在多个表中时，那么就可以创建视图来将这些满足条件的行和列组织到一个表中，而不需要修改表的属性，甚至创建新的表。这样不仅简化了用户的操作，还可以提高数据的逻辑独立性，实现数据的共享和保密。

1. 用 SQL 命令创建视图

1）语法格式

```
CREATE [OR REPLACE] VIEW [<模式名>.]<视图名>[(<列名> {,<列名>})] AS <查询说明> [WITH [LOCAL|CASCADED]CHECK OPTION]|[WITH READ ONLY];
```

其中，各子句说明如下：

```
<查询说明>::=<表查询> | <表连接>
<表查询>::=<子查询表达式>[ORDER BY 子句]
```

WITH CHECK OPTION 参数指明向该视图中插入或修改数据时，插入行或更新行的数据必须满足视图定义中<查询说明>所指定的条件，如果不带该选项，则插入行或更新行的数据不必满足视图定义中<查询说明>所指定的条件。

2）应用举例

【例 4-42】创建基于单表的视图。对 VENDOR 表创建一个视图，名为 VENDOR_EXCELLENT，保存信誉等级为 1 的供应商，列名有 VENDORID、ACCOUNTNO、NAME、ACTIVEFLAG、CREDIT。

（1）创建视图。

```
SQL> CONN SYSDBA/SYSDBA;
SQL> CREATE VIEW purchasing.vendor_excellent AS
SELECT vendorid, accountno, name, activeflag, credit
FROM purchasing.vendor
WHERE credit = 1;
```

由于视图列名与查询说明中 SELECT 后的列名相同，所以视图名后的列名可省略。

运行该语句，AS 后的查询语句并未执行，系统只是将所定义的<视图名>及<查询说明>送入数据字典保存。对用户来说，就像在数据库中已经有 VENDOR_EXCELLENT 表一样。

（2）查询视图数据：

```
SQL> SELECT vendorid, name FROM purchasing.vendor_excellent;
行号          VENDORID      NAME
---------- ---------- --------------------------------------------------
1             3             北京十月文艺出版社
2             4             人民邮电出版社
3             5             清华大学出版社
4             6             中华书局
5             7             广州出版社
6             8             上海出版社
7             9             21 世纪出版社
8             10            外语教学与研究出版社
9             11            机械工业出版社
10            12            文学出版社
10 rows got
```

【例 4-43】创建基于多表的视图。创建 SALESPERSON_INFO 视图，基于 SALES.SALESPERSON 表、RESOURCES.EMPLOYEE 表和 PERSON.PERSON 表，用来保存销售人员的信息，列名有 SALESPERSONID、TITLE、NAME、SALESLASTYEAR。

（1）创建视图：

```
SQL> CREATE VIEW sales.salesperson_info AS
SELECT t1.salespersonid, t2.title, t3.name, t1.saleslastyear
FROM sales.salesperson t1,   resources.employee t2,   person.person t3
WHERE t1.employeeid = t2.employeeid   AND   t2.personid = t3.personid;
```

（2）查询视图数据：

```
SQL> SELECT * FROM sales.salesperson_info;
行号          SALESPERSONID        TITLE      NAME     SALESLASTYEAR
---------- ------------ -------- ---- ----------------------------------------------------------------
  1               1              销售代表     郭艳          10.00
  2               2              销售代表     孙丽          20.00
```

为了减少数据冗余，由表中数据经各种计算统计出的数据一般不再存储，但这样的计算数据往往又要经常使用，这时可将它们定义成视图中的数据。

【例 4-44】创建用于统计的视图。创建 VENDOR_STATIS 视图，在 PRODUCT_VENDOR 表的基础上，统计厂家产品数量。

（1）创建视图：

```
SQL> CREATE VIEW production.vendor_statis(vendorid, product_count) AS
SELECT vendorid, count(productid)
FROM production.product_vendor
GROUP BY vendorid
ORDER BY vendorid;
```

在该语句中，由于 SELECT 后出现了集函数 COUNT(PRODUCTID)，不属于单纯的字段名，所以视图中的对应列必须重新命名，即在<视图名>后明确说明视图的各个字段名。

（2）查询视图数据：

```
SQL>SELECT * FROM production.vendor_statis;
行号        VENDORID       PRODUCT_COUNT
---------- ----------------- ------------------------------
1               5                  1
2               6                  2
3               7                  1
4               8                  1
5               9                  1
6               10                 1
7               11                 1
7 rows got
```

【例 4-45】向视图中插入数据。给 VENDOR_EXCELLENT 视图插入一条数据，ACCOUNTNO 的值为'00'，NAME 的值为 '北京大学出版社'，ACTIVEFLAG 的值为 1。

（1）向视图插入数据：

```
SQL> INSERT INTO purchasing.vendor_excellent(ACCOUNTNO, NAME, ACTIVEFLAG,CREDIT)
VALUES('00', '北京大学出版社',1,1);
```

（2）查询视图数据：

```
SQL> SELECT vendorid, name FROM purchasing.vendor_excellent WHERE vendorid > 10;
行号        VENDORID       NAME
---------- ----------- ---------------------------------------
1               11              机械工业出版社
2               12              文学出版社
3               13              北京大学出版社
```

这个例子说明可以向视图插入数据。

（3）查询源表数据：

```
SQL> SELECT VENDORID, NAME FROM PURCHASING.VENDOR WHERE VENDORID > 10;
行号        VENDORID       NAME
---------- ----------- ---------------------------------------
1               11              机械工业出版社
```

2	12	文学出版社
3	13	北京大学出版社

查询结果说明，向视图插入数据时，最终插入到视图对应的源表中，反之，它印证视图中并没有真正存储数据。除了可以向视图插入数据外，还可以向视图更新和删除数据。但是对基于统计的视图插入数据时，如果视图查询中包含下列结构——连接、集合运算符、GROUP BY 子句，则该视图上不能进行插入、修改和删除操作。

为了防止用户通过视图更新基表数据，无意或故意更新了不属于视图范围内的源表数据，在视图定义语句的子查询后提供了可选项 WITH CHECK OPTION，表示向该视图中插入或修改数据时，要保证插入行或更新行的数据满足视图定义中<查询说明>所指定的条件。

2. 用管理工具创建视图

【例 4-46】创建基于单表的视图。基于 DMHR 模式的 CITY 表，建立华南地区城市视图 SCCITY。

步骤 1：右击 REGION 表，在弹出的快捷菜单中选择“浏览数据”选项，在管理工具的右方显示 REGION 表的数据，如图 4-44 所示，华南地区的 REGION_ID 字段值为 3。

图 4-44　浏览数据

步骤 2：在图 4-45 中右击“视图”节点，在弹出的快捷菜单中选择“新建视图”选项，弹出“新建视图”对方框，如图 4-46 所示。

步骤 3：在图 4-46 中，在“视图名”文本框中输入 SCCITY，单击“查询设计器”按钮，弹出“查询设计”对话框，如图 4-47 所示。

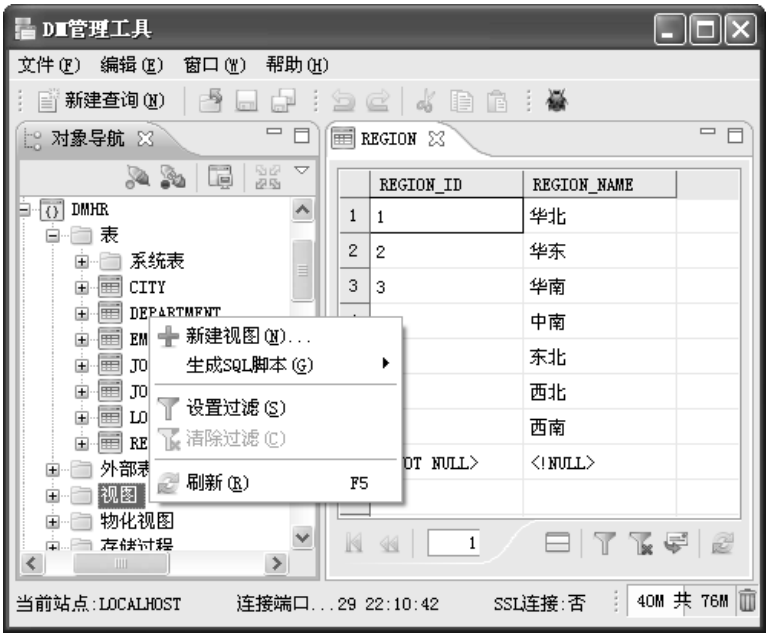

图 4-45　新建视图

图 4-46　视图常规参数设定

步骤 4：在图 4-47 中，在“目标对象”右侧单击“+”按钮，弹出“对象选择”对话框，如图 4-48 所示，在“表”选项卡中选择模式为 DMHR，并选中 CITY 表。

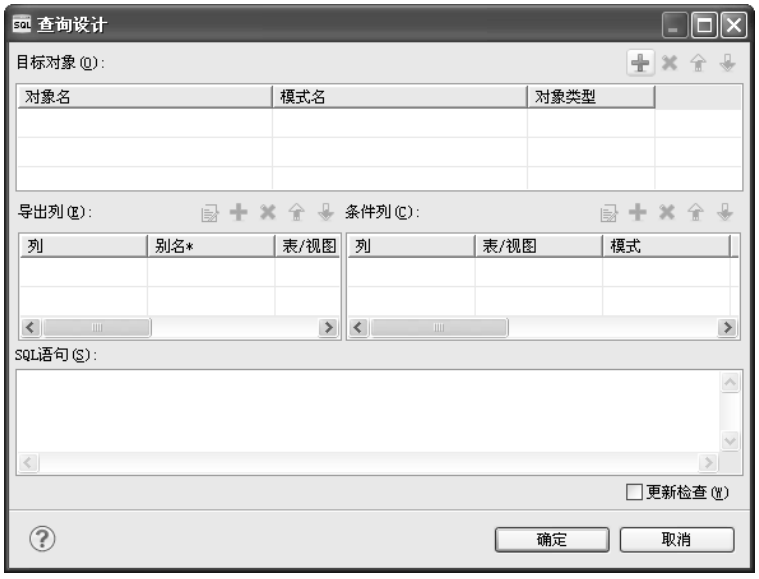

图 4-47　查询设计

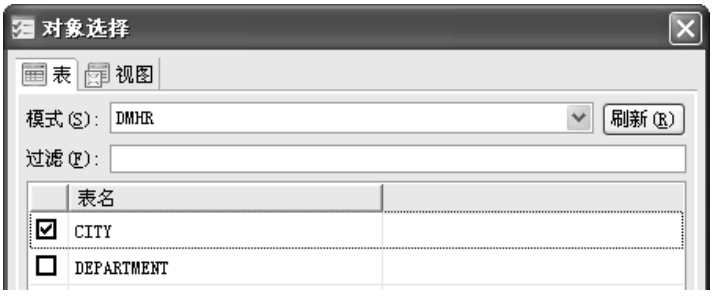

图 4-48　选择查询表

步骤 5：在图 4-47 中，在“导出列”右侧单击“+”按钮，弹出“列选择”对话框，如图 4-49 所示，选中全部列。

图 4-49　选择导出字段

步骤 6：在图 4-47 中，在“条件列”右侧单击“+”按钮，弹出“列选择”对话框，如图 4-50 所示，选中 REGION_ID 字段。返回“查询设计”对话框，将 REGION_ID 列的比较符设置为“=”，比较值设置为“3”。

图 4-50　选择条件字段

步骤 7：经过上述设置，查询设计会自动构建视图的查询语句，如图 4-51 所示。单击“确定”按钮，返回“新建视图”对话框，如图 4-52 所示，单击“确定”按钮，完成视图的创建过程。

图 4-51　自动设计查询的 SQL 语句

步骤 8：右击 SCCITY 视图，在弹出的快捷菜单中选择“浏览数据”选项，在管理工具的右侧显示 SCCITY 视图的数据，如图 4-53 所示。

图 4-52　视图设计结果

图 4-53　浏览视图数据

【例 4-47】创建基于多表的视图。在 DMHR 模式下，基于 REGION 表和 CITY 表创建华南地区城市视图 SCCITY2。

步骤 1：在“查询设计”对话框中，目标对象选择 CITY 表和 REGION 表，如图 4-54 所示。

步骤 2：选择导出列时，选择 CITY 表的所有字段，如图 4-55 所示。

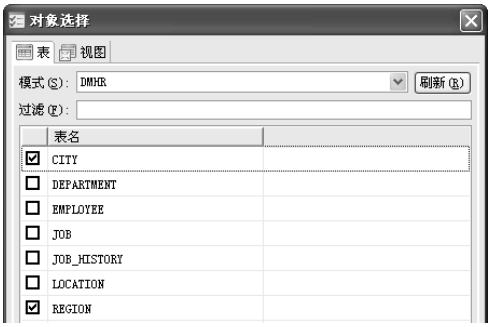

图 4-54　选择查询表

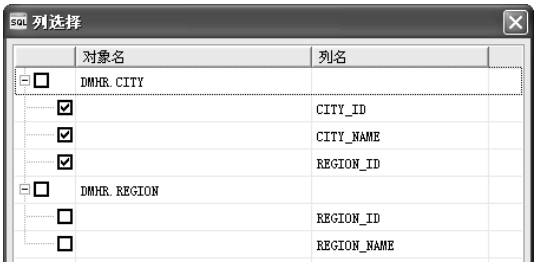

图 4-55　选择视图字段

步骤 3：选择条件字段时，增加 REGION 表的 REGION_NAME 字段，比较符为“=”，比较值为'华南'。增加 CITY 表的 REGION_ID 字段，比较符为“=”，比较值为 REGION.REGION_ID。最终的条件字段结果如图 4-56 所示。

图 4-56　选择条件字段并编辑条件

步骤 4：视图创建完成后，浏览数据，结果如图 4-57 所示。

图 4-57　浏览数据

4.4.2　删除视图

由于视图中没有真正存放数据，因此删除视图也不会真正删除数据。一个视图删除后，会影响到基于该视图的其他视图，因此删除视图也不是随意的。

1. 用 SQL 命令删除视图

一个视图本质上是基于其他源表或视图上的查询，我们把这种对象间的关系称为依赖。用户在创建视图成功后，系统还隐式地建立了相应对象间的依赖关系。在一般情况下，当一个视图不被其他对象依赖时可以随时删除视图。

1）*语法格式*

删除视图的 SQL 命令如下：

```
DROP VIEW [<模式名>.]<视图名> [RESTRICT | CASCADE];
```

视图删除有两种方式，即 RESTRICT 和 CASCADE 方式，其中 RESTRICT 为默认值。当设置 dm.ini 中的参数 DROP_CASCADE_VIEW 值为 1 时，如果在该视图上建有其他视图，则必须使用 CASCADE 参数才可以删除所有建立在该视图上的视图，否则删除视图的操作不会成功；当设置 dm.ini 中的参数 DROP_CASCADE_VIEW 值为 0 时，RESTRICT 和 CASCADE 方式都会成功，且只会删除当前视图，不会删除建立在该视图上的视图。

2）*应用举例*

【例 4-48】删除 SCCITY 视图。

```
SQL> DROP VIEW dmhr.sccity;
```

2. 用管理工具删除视图

【例 4-49】删除 DMHR 模式下的 SCCITY2 视图。

步骤 1：在图 4-58 中右击 SCCITY2 视图，在弹出的快捷菜单中选择“删除”选项，

弹出“删除对象”对话框，如图 4-59 所示。

图 4-58　删除视图

图 4-59　确认删除视图

步骤 2：在图 4-59 中确认 SCCITY2 相关设定无误后，单击“确定”按钮，删除 SCCITY2 视图。

4.4.3　创建物化视图

物化视图是从一个或几个源表导出的表，同视图相比，它存储了导出表的真实数据。当源表中的数据发生变化时，物化视图所存储的数据将变得陈旧，用户可以通过手动刷新或自动刷新来对数据进行同步。

1）物化视图的一般限制

（1）物化视图不能包含集合操作。

（2）物化视图定义不能含有垂直分区表。

（3）对物化视图日志、物化视图只能进行查询和创建索引，不支持插入、删除、更新、

MERGE INTO 和 TRUNCATE。

（4）同一表上最多允许建立 127 个物化视图。

（5）包含物化视图的普通视图及游标是不能更新的。

（6）如果对某明细表进行了 TRUNCATE 操作，那么依赖于它的物化视图必须先进行一次完全刷新后才可以使用快速刷新。

2）物化视图的分类

依据物化视图定义中查询语句的不同，物化视图分为以下五种。

（1）SIMPLE：无 GROUP BY，无聚集函数，无连接操作。

（2）AGGREGATE：仅包含 GROUP BY 聚集函数。

（3）JOIN：仅包含多表连接。

（4）Sub-Query：仅包含子查询。

（5）COMPLEX：除上述四种以外的物化视图类型。

用户可以通过查看系统视图 SYS.USER_MVIEWS 的 MVIEW_TYPE 列来了解所定义物化视图的分类。

3）快速刷新通用约束

（1）快速刷新物化视图要求每个基表都包含物化视图日志，并且物化视图日志的创建时间不得晚于物化视图的最后刷新时间。

（2）不能含有不确定性函数，如 SYSDATE 或 ROWNUM。

（3）不能含有大字段类型。

（4）查询项不能含有分析函数。

（5）查询不能含有 HAVING 子句。

（6）不能包含 ANY、ALL 及 NOT EXISTS。

（7）不能含有层次查询。

（8）不能在多个站点含有相关表。

（9）同一张表上最多允许建立 127 个快速刷新的物化视图。

（10）不能含有 UNION、UNION ALL、MINUS 等集合运算。

（11）不能含有子查询。

（12）只能基于普通表（视图、外部表、派生表等不支持）。

（13）WITH PRIMARY KEY 时物化视图定义里只能是单表并且日志表里有 PK，多表时必须是 WITH ROWID 并且日志表里有 ROWID，WITH ROWID 刷新时定义物化视图可以是单表，前提是日志表里有 ROWID。

（14）当 WITH ROWID 的快速刷新单表和多表时需要一一选择 ROWID 并给出别名。

（15）单表 WITH PK 刷新时，视图定义中必须包含所有的 PK 列。

（16）如果日志定义中没有 WITH PRIMARY KEY 而扩展列又包含了，那么 DM 7 认为其和建立日志时指定的 WITH PRIMARY KEY 效果相同。也就是说，基于这个日志建立 WITH PK 的快速刷新物化视图是允许的。

（17）DM 7 目前仅支持简单类型物化视图的快速刷新。

1. 用 SQL 命令创建物化视图

1）语法格式

创建物化视图的 SQL 命令格式如下：

```
CREATE MATERIALIZED VIEW [<模式名>.]<物化视图名>[(<列名>{,<列名>})][BUILD IMMEDIATE|BUILD DEFERRED][<STORAGE 子句>][<物化视图刷新选项>][<查询改写选项>]AS<查询说明>
```

其中，各子句说明如下：

```
<查询说明>::= <表查询> | <表连接>
<表查询>::=<子查询表达式>[ORDER BY 子句]
<物化视图刷新选项> ::= REFRESH <刷新选项> {<刷新选项>} | NEVER REFRESH
<刷新选项> ::= [FAST | COMPLETE | FORCE] [ON DEMAND | ON COMMIT ] [START WITH datetime_expr | NEXT datetime_expr] [WITH PRIMARY KEY | WITH ROWID]
<查询改写选项>::= [DISABLE | ENABLE] QUERY REWRITE
<datetime_expr>::= SYSDATE[+数值常量]
```

语法中的参数说明见表 4-11。

表 4-11 参数说明

参 数	说 明
BUILD IMMEDIATE BUILD DEFERRED	BUILD IMMEDIATE 为立即填充数据，默认为立即填充； BUILD DEFERRED 为延迟填充，使用这种方式要求第一次刷新必须为 COMPLETE，即完全刷新
刷新模式	FAST，根据相关表上的数据更改记录进行增量刷新。普通 DML 操作生成的记录存在于物化视图日志。使用 FAST 刷新之前，必须先建好物化视图日志。 COMPLETE，通过执行物化视图的定义脚本进行完全刷新。 FORCE，默认选项。当快速刷新可用时采用快速刷新，否则采用完全刷新
刷新时机	ON COMMIT，在相关表上事务提交时进行快速刷新，这会增加 COMMIT 完成的时间。DM 7 目前仅语法支持 ON COMMIT，实际功能并未实现。 约束： 含有对象类型的不支持； 包含远程表的不支持。
刷新时机	START WITH … NEXT，START WITH 用于指定首次刷新物化视图的时间，NEXT 指定自动刷新的间隔；如果省略 START WITH，则首次刷新时间为当前时间加上 NEXT 指定的间隔；如果指定 START WITH 省略 NEXT，则物化视图只会刷新一次；如果二者都未指定，则物化视图不会自动刷新。 ON DEMAND，由用户通过 REFRES 语法进行手动刷新。如果指定了 START WITH 和 NEXT 子句，则没有必要指定 ON DEMAND。 NEVER REFRESH，物化视图从不进行刷新。可以通过 ALTER MATERALIZED VIEW <物化视图名> FRESH 进行更改

（续表）

参　数	说　明
刷新选项	WITH PRIMARY KEY，默认选项。除 WITH ROWID 之外都要选择此选项。必须含有 PRIMARY KEY 约束，选择列必须直接含有所有的 PRIMARY KEY（UPPER(col_name)的形式不可接收）。不能含有对象类型。 WITH ROWID，必须基于单表，并且不能包含如下内容： DISTINCT 或聚集函数； GROUP BY 或 CONNECT BY 子句； 子查询； 连接； 集合运算； 不能含有对象类型。 如果使用 WITH ROWID 的同时使用快速刷新，则必须将 ROWID 提取出来，和其他列名一起，以别名的形式显示
查询改写选项	ENABLE QUERY REWRITE，允许物化视图用于查询改写。 DISABLE QUERY REWRITE，禁止物化视图用于查询改写。 目前 DM 7 仅语法支持查询改写选项，实际功能未实现
时间表达式	SYSDATE[+数值常量]或日期间隔

2）应用举例

【例 4-50】对 VENDOR 表创建一个物化视图，名为 MV_VENDOR_EXCELLENT，保存信誉等级为 1 的供应商，列名有 VENDORID、ACCOUNTNO、NAME、ACTIVEFLAG、CREDIT。不允许查询改写，依据 ROWID 刷新且刷新间隔为一天。

（1）作业准备。

这个例子要求依据 ROWID 刷新且刷新间隔为一天，这涉及到作业的内容，请按照第 6 章 6.1 节“5.作业准备”的要求完成作业准备工作。

（2）创建物化视图。

```
SQL> CREATE MATERIALIZED VIEW purchasing.mv_vendor_excellent
REFRESH WITH ROWID
START WITH SYSDATE NEXT SYSDATE + 1
AS
SELECT VENDORID, ACCOUNTNO, NAME, ACTIVEFLAG, CREDIT
FROM PURCHASING.VENDOR
WHERE CREDIT = 1;
```

运行该语句后，在达梦数据库服务器将得到以下内容。

① 物化视图：MV_VENDOR_EXCELLENT。

② 定时刷新的物化视图作业：MJOB_REFRESH_MVIEW_1873，用于定时刷新。

（3）查询物化视图的数据：

```
SQL> SELECT VENDORID, NAME, ACTIVEFLAG, CREDIT FROM PURCHASING.MV_ VENDOR_EXCEL;
```

```
行号   VENDORID    NAME                    ACTIVEFLAG   CREDIT
---- ---------- -------------------- ---------- ---------- ------- ------- ------- ------- -------
1          3          北京十月文艺出版社              1            1
2          4          人民邮电出版社                  1            1
3          5          清华大学出版社                  1            1
4          6          中华书局                        1            1
5          7          广州出版社                      1            1
6          8          上海出版社                      1            1
7          9          21 世纪出版社                   1            1
8          10         外语教学与研究出版社            1            1
9          11         机械工业出版社                  1            1
10         12         文学出版社                      1            1
11         13         北京大学出版社                  1            1
11 rows got
```

【例 4-51】对 VENDOR 表创建一个物化视图，名为 MV_VENDOR_ EXCELLENT2，保存信誉等级为 1 的供应商，列名有 VENDORID、ACCOUNTNO、NAME、ACTIVEFLAG、CREDIT。依据 ROWID 使用快速刷新。

```
SQL> CREATE MATERIALIZED VIEW    purchasing.mv_vendor_excellent2
REFRESH WITH ROWID
ON COMMIT
AS
SELECT vendorid, accountno, name, activeflag, credit, rowid AS x
FROM purchasing.vendor
WHERE credit = 1;
```

运行该语句后，在服务器中将得到物化视图：MV_VENDOR_EXCELLENT2。

2. 用管理工具创建物化视图

【例 4-52】对 DMHR 模式 CITY 表创建一个物化视图，名为 MV_SCCITY，保存华南地区城市名，使用 CITY 的全部字段。不允许查询改写，依据 ROWID 使用快速刷新。

步骤 1：在图 4-60 中右击“物化视图”下的“日志表”节点，在弹出的快捷菜单中选择“新建日志表”选项，弹出“新建物化视图日志表”对话框，如图 4-61 所示，设置表名为“CITY”，其他采用默认参数，单击“确定”按钮，完成物化视图日志表的创建任务。

步骤 2：在图 4-62 中右击“物化视图”节点，在弹出的快捷菜单中选择“新建物化视图”选项，弹出“新建物化视图”对话框，如图 4-63 所示。

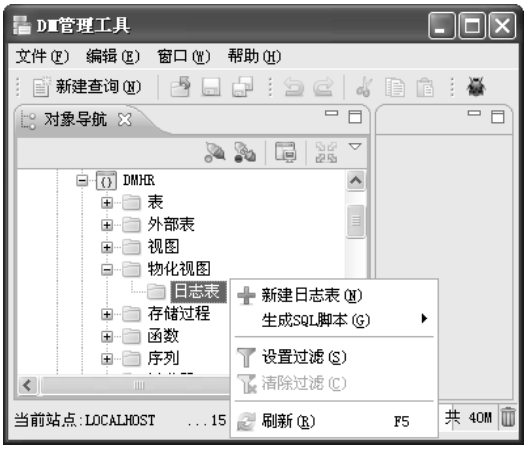

图 4-60　新建物化视图日志表

图 4-61　设置物化视图日志表参数

图 4-62　新建物化视图

图 4-63 “新建物化视图”对话框

步骤 3：在图 4-63 的“常规”页面中，设置视图名为 MV_SCCITY，设置子查询如下。

```
SELECT  *  FROM  DMHR.CITY
WHERE  REGION_ID=3;
```

步骤 4：继续设置刷新属性，如图 4-64 所示，BUILD 选项设置为“BUILD IMMEDIATE”，QUERY 选项设置为“DISABLE QUERY REWRITE”，REFRESH 选项设置为“REFRESH”，并选中“ON COMMIT”、“WITH ROWID”、“FAST”单选按钮。单击“确定”按钮，完成构建物化视图过程。

图 4-64 设置刷新参数

步骤 5：为了检验 MV_SCCITY，对其进行查询，查询结果如图 4-65 所示。

图 4-65　查询物化视图数据

4.4.4　修改物化视图

1. 用 SQL 命令修改物化视图

（1）语法格式。

修改物化视图的 SQL 命令格式如下：

```
ALTER MATERIALIZED VIEW [<模式名>.]<物化视图名> [<物化视图刷新选项>] [<查询改写选项>]
```

使用者必须是该物化视图的拥有者或者拥有 ALTER ANY MATERIALIZED VIEW 系统权限。

（2）应用举例。

【例 4-53】修改物化视图 MV_VENDOR_EXCELLENT，使之可以查询改写。

```
SQL> ALTER MATERIALIZED VIEW PURCHASING.MV_VENDOR_EXCELLENT ENABLE QUERY REWRITE;
```

【例 4-54】修改物化视图 MV_VENDOR_EXCELLENT 为完全刷新。

```
SQL> ALTER MATERIALIZED VIEW PURCHASING.MV_VENDOR_EXCELLENT REFRESH COMPLETE;
```

2. 用管理工具修改物化视图

【例 4-55】修改物化视图 MV_VENDOR_EXCELLENT2，采用完全刷新并可以用于查询改写。

右击 PURCHASING 模式下的物化视图 MV_VENDOR_EXCELLENT2，在弹出的快捷菜单中选择“修改”选项，弹出修改物化视图对话框，在 QUERY 选项中选择“ENABLE

QUERY REWRITE”选项，在 REFRESH 选项中选择“COMPLETE”，其他保持不变，如图 4-66 所示。

图 4-66　修改物化视图

4.4.5　删除物化视图

由于物化视图中存储了从表中导出的数据，因此删除物化视图时自然会删除物化视图的数据，但是不会删除源表中的数据。

1. 用 SQL 命令删除物化视图

（1）语法格式。

```
DROP MATERIALIZED VIEW [<模式名>.]<物化视图名>;
```

删除物化视图时会清除物化视图、物化视图表及定时刷新作业（如果存在的话）；物化视图删除后，用户在其上的权限也均自动取消，以后系统中再创建的同名物化视图，是与它毫无关系的物化视图；不能直接删除物化视图表对象。

（2）应用举例。

【例 4-56】删除物化视图 MV_VENDOR_EXCELLENT。

```
SQL> DROP MATERIALIZED VIEW purchasing.mv_vendor_excellent;
```

2. 用管理工具删除物化视图

【例 4-57】删除物化视图 MV_VENDOR_EXCELLENT2。

步骤 1：在图 4-67 中右击 MV_VENDOR_EXCELLENT2，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-68 所示。

步骤 2：在图 4-68 中，确认删除的 MV_VENDOR_EXCELLENT2 无误后，单击“确定”按钮，删除该物化视图。

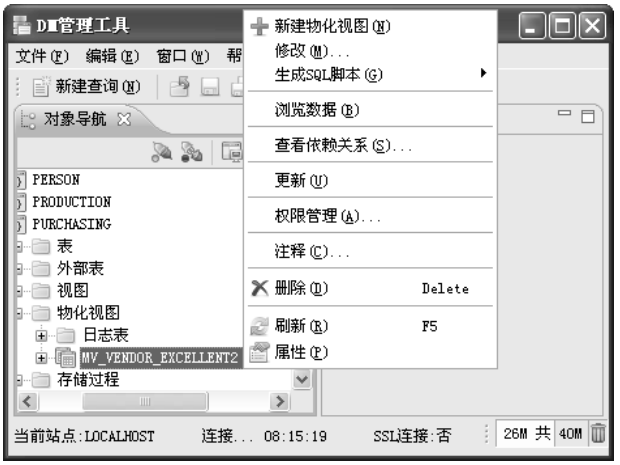

图 4-67　删除物化视图

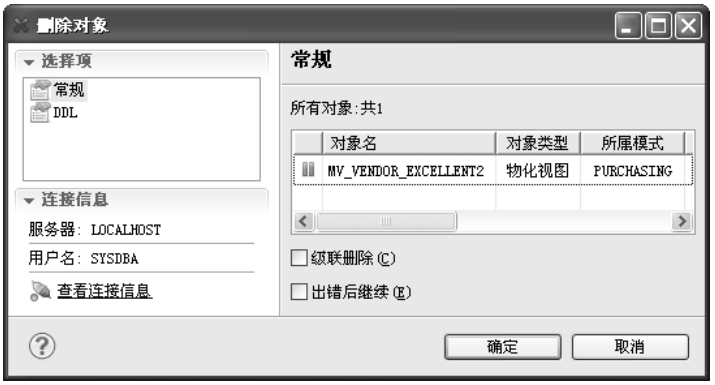

图 4-68　确认删除

4.5　索引管理

索引是与表相关的一种结构，它能使对应于表的 SQL 语句执行得更快，因为索引能更快地定位数据。DM 7 索引能提供访问表的数据的更快路径，可以不用重写任何查询而使用索引，其查询结果与不使用索引是一样的，但速度更快。

DM 7 提供了几种最常见类型的索引，对不同场景有不同的功能。

（1）聚集索引：每一个普通表有且只有一个聚集索引。

（2）唯一索引：索引数据根据索引键唯一。

（3）函数索引：包含函数/表达式的预先计算的值。

（4）位图索引：对低基数的列创建位图索引。

（5）位图连接索引：针对两个或者多个表连接的位图索引，主要用于数据仓库中。

（6）全文索引：在表的文本列上创建的索引。

索引需要存储空间。创建或删除一个索引，不会影响基本表、数据库应用或其他索引。

一个索引可以对应数据表的一个或多个字段，对每个字段设置索引结果排序方式，默认为按字段值递增排序（ASC），也可以指定为递减排序（DESC）。

当插入、更改和删除相关的表的行时，DM 7 会自动管理索引。如果删除索引，所有的应用仍继续工作，但访问数据的速度会变慢。

索引可以提高数据的查询效率，但也需要注意，索引会降低某些命令的执行效率，如 INSERT、UPDATE、DELETE 的性能，因为 DM 不仅要维护基表数据还要维护索引数据。

4.5.1 创建常用索引

1. 用 SQL 命令创建索引

1）语法格式

创建索引的 SQL 命令格式如下：

```
CREATE [OR REPLACE] [CLUSTER|NOT PARTIAL][UNIQUE | BITMAP| SPATIAL] INDEX <索引名> ON [<模式名>.]<表名>(<索引列定义>{,<索引列定义>})  [GLOBAL] [<STORAGE 子句>] [NOSORT] [ONLINE];
```

通过该 SQL 命令可以创建普通索引、聚集索引、唯一索引、位图索引等。

ON 关键字表示在哪个表的哪个字段上建立索引，字段的类型不能是多媒体类型。在字段后面指定索引排序方式，ASC 表示递增排序，DESC 表示递减排序，默认为递增排序。

STORAGE 关键字设置索引存储的表空间，默认与对应表的表空间相同。

2）应用举例

【例 4-58】在单个字段上建立普通索引。以 SYSDBA 用户给 PURCHASING 模式的 VENDOR 表的 VENDORID 字段建立普通索引，索引名为 S1。

```
SQL> CREATE INDEX s1 ON purchasing.vendor(vendorid);
```

【例 4-59】在多个字段上建立唯一索引。以 SYSDBA 用户给 PURCHASING 模式的 VENDOR 表的 ACCOUNTNO 和 NAME 字段建立唯一索引，索引名为 S2。

```
SQL> CREATE UNIQUE INDEX s2 ON purchasing.vendor(accountno, name);
```

【例 4-60】在单个字段上建立函数索引。SYSDBA 用户给 DMHR 模式下的 CITY 表的 CITY_ID 字段建立 LOWER()函数索引，索引名为 CITY_LOWER。

（1）创建函数索引。

```
SQL> CREATE INDEX city_lower ON dmhr.city(LOWER(city_id));
```

（2）利用函数索引查询数据。

```
SQL> SELECT * FROM dmhr.city WHERE LOWER(city_id) = 'wh';
行号        CITY_ID     CITY_NAME     REGION_ID
---------- ------- --------- ----------------------------------------------------
1           WH          武汉          4
```

这个例子说明，创建函数索引后，达梦数据库已经将针对 CITY 表的 CITY_ID 字段的 LOWER 函数计算结果都存储起来了。如果某个查询中包含针对 CITY_ID 字段的 LOWER 函数，DM 在执行查询时不用再做函数运算，而是直接利用函数索引中存储的计算结果，这可以提高查询的速度。

【例 4-61】在低基数字段上建立位图索引。以 SYSDBA 用户给 DMHR 模式下的 EMPLOYEE 表的 JOB_ID 字段建立位图索引。

（1）创建位图索引。

```
SQL> CREATE BITMAP INDEX dmhr.empjob_idx ON dmhr.employee(job_id);
```

（2）利用位图索引查询数据。

```
SQL> SELECT employee_id, employee_name, salary FROM dmhr.employee WHERE job_id = 21;
行号     EMPLOYEE_ID       EMPLOYEE_NAME     SALARY
---------- ----------- ------------- ----------------------------------------------------------
1          1002              程擎武            9000.00
2          2002              常鹏程            5000.00
3          3002              强洁芳            10000.00
4          4002              张晓中            6000.00
5          5002              郑成功            6000.00
6          6002              商林玉            5000.00
7          7002              戴慧华            6000.00
8          8002              罗利平            5000.00
9          9002              刘春天            5000.00
10         10002             王岳荪            5000.00
11         11002             蔡玉向            5000.00
11 rows got
```

低基数字段指字段取值比较少，即该字段值相同的记录有很多条，该字段值对全表记录的区分度不大，基于该字段的查询效率低。例如，EMPLOYEE 表中 JOB_ID（职务编号）字段就是典型的低基数字段，只有 16 个值，相同 JOB_ID 指有很多条记录。对低基数字段建立普通索引对查询效率提高不大，建立位图索引则可以大大提高查询效率。

对 JOB_ID 建立位图索引后，DM 7 会按 JOB_ID 的值的个数（16 个）建立 16 个向量，以“21”为例，在 EMPLOYEE 全表中 JOB_ID 值为 21 的记录对应为 1，其他值对应为 0，这样就对全表建立了一个向量（0,1,0,0,1,1，…）。以此类推，建立其他 15 个向量。当以 JOB_ID=21 为查询条件时，直接查看“21”对应向量（0,1,0,0,1,1，…），向量元素值为 1 就是被查找的记录，查询效率大大提高了。

3）附加说明

（1）位图连接索引名称的长度限制为事实表名的长度+索引名称长度+6<128。

（2）仅支持普通表、LIST 表和 HFS 表。

（3）WHERE 条件只能是列与列之间的等值连接，并且必须含有所有表。

（4）位图连接索引表（命名为 BMJ$_索引名）仅支持 select 操作，不支持 insert、delete、

update、alter、drop 等操作。

2. 用管理工具创建索引

【例 4-62】在单个字段上创建普通索引。使用 SYSDBA 用户给 DMHR 模式的 EMPLOYEE_NAME 字段创建索引，索引名为 INDEX_EMPNAME。

步骤 1：在图 4-69 中右击“索引”节点，在弹出的快捷菜单中选择“新建索引”选项，弹出“新建索引”对话框，如图 4-70 所示。

步骤 2：在图 4-70 中，设置索引名称为“INDEX_EMPNAME”，将 EMPLOYEE_NAME 字段加入到索引列中，其他参数保持不变。单击“确定”按钮，完成索引创建。

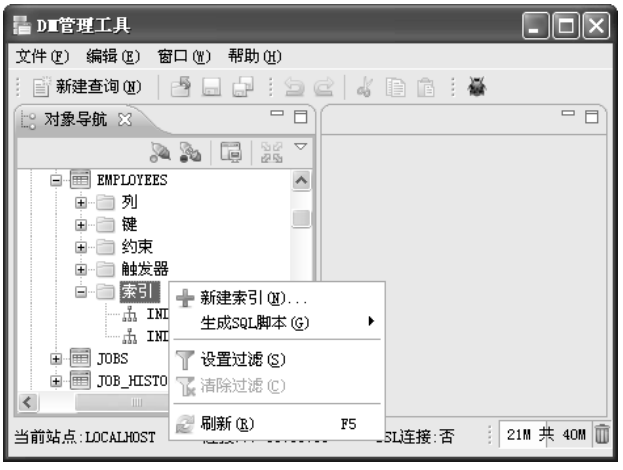

图 4-69　新建索引

图 4-70　设置索引参数

4.5.2　删除常用索引

索引是数据表的外在部分，删除索引不会删除表的任何数据，也不改变表的使用方式，只是会影响表数据的查询速度。

1. 用 SQL 命令删除索引

1）语法格式

删除索引的 SQL 命令格式如下：

```
DROP INDEX [<模式名>.]<索引名>;
```

删除索引的用户应拥有 DBA 权限或是该索引所属基表的拥有者。

2）应用举例

【例 4-63】具有 DBA 权限的用户删除 PURCHASING 模式下的 S1 索引。

（1）删除索引。

```
SQL> DROP INDEX purchasing.s1;
```

（2）查询 S1 索引对应的 VENDOR 表的数据：

```
SQL> SELECT vendorid, name FROM purchasing.vendor WHERE vendorid > 10;
行号        VENDORID    NAME
---------- ---------- ----------------------------------------
1           11          机械工业出版社
2           12          文学出版社
3           13          北京大学出版社
```

这个例子表明，删除索引不会删除表的数据，也不会改变表的使用方式，只会降低数据查询速度。

2. 用管理工具删除索引

【例 4-64】删除 DMHR 模式下 EMPLOYEES 表的 INDEX_EMPNAME 索引。

步骤 1：在图 4-71 中，右击 EMPLOYEES 表的 INDEX_EMPNAME 索引，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-72 所示。

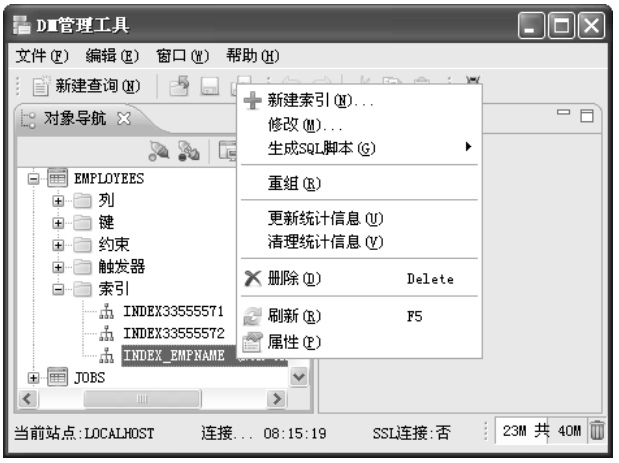

图 4-71　删除索引

图 4-72　确认删除

步骤 2：在图 4-72 中，确认 INDEX_EMPNAME 无误后，单击“确定”按钮删除该索引。

4.5.3　创建位图连接索引

位图连接索引是一种通过连接提高海量数据查询效率的有效方式，主要用于数据仓库中。它是针对两个或者多个表连接的位图索引，同时保存了连接的位图结果。对于字段中的每一个值，该索引保存了索引表中对应行的 ROWID。

1. 用 SQL 命令创建位图连接索引

1）语法格式

创建位图连接索引的命令格式如下：

```
CREATE [OR REPLACE] BITMAP INDEX <索引名> ON bitmap_join_index_clause [<STORAGE 子句>];
```

其中，各子句说明如下：

```
bitmap_join_index_clause::=[<模式名>.]<表名>(<索引列定义>{,<索引列定义>})FROM [<模式名>.]<基表名>[别名]{,[<模式名>.]<基表名>[别名]} WHERE <条件表达式>;
<索引列定义>::= [[<模式名>.]<表名|别名>]<索引列表达式>[ASC|DESC]
```

在 ON 关键字之后，是事实表，括号中的字段可以是事实表的字段，也可以是维度表的字段。在 FROM 关键字之后是参与连接的表名。在 WHERE 关键字之后是表连接条件，连接字段必须是维度表的主键或具有唯一性约束。

2）应用举例

【例 4-65】对 SALES 模式下的 SALESORDER_HEADER 表与 CUSTOMER 表之间创建位图连接索引，索引名称为 SALES_CUSTOMER_ NAME_IDX，索引列为 CUSTOMER.PERSONID。

（1）创建位图连接索引：

```
SQL> CREATE BITMAP INDEX sales.sales_customer_name_idx
ON sales.salesorder_header(sales.customer.personid)
FROM sales.customer, sales.salesorder_header
WHERE sales.customer.customerid = sales.salesorder_header.customerid;
```

（2）执行查询：

```
SQL> SELECT total from sales.customer, sales.salesorder_header
WHERE sales.customer.customerid = sales.salesorder_header.customerid
and sales.customer.personid = '16';
行号        TOTAL
---------- --------------------
1           36.90
2           36.90
```

上述例子说明，建立了位图连接索引之后，查询还是传统的表连接查询语句，DM 7 在执行这个查询时，不需要进行 SALESORDER_HEADER 表与 CUSTOMER 表的连接操作，因为位图连接索引中已经存储了两个表连接后的数据，可以大大提高查询速度。

3）附加说明

（1）位图连接索引名称的长度限制为事实表名的长度+索引名称长度+6<128。

（2）位图连接索引仅支持普通表、LIST 表和 HFS 表。

（3）位图连接索引 WHERE 条件只能是字段与字段之间的等值连接，并且必须含有所有表。

（4）位图连接索引表（命名为 BMJ$_索引名）仅支持 select 操作，不支持 insert、delete、update、alter、drop 等操作。

2. 用管理工具创建位图连接索引

【例 4-66】在 DMHR 模式下的 CITY 表与 REGION 表之间创建位图连接索引，索引名称为 REGOIN_NAME_IDX，索引列为 REGION.REGOIN_NAME。

步骤 1：在图 4-73 中右击 DMHR 模式下的 CITY 表的索引，在弹出的快捷菜单中选择“新建索引”选项，弹出“新建索引”对话框，如图 4-74 所示。

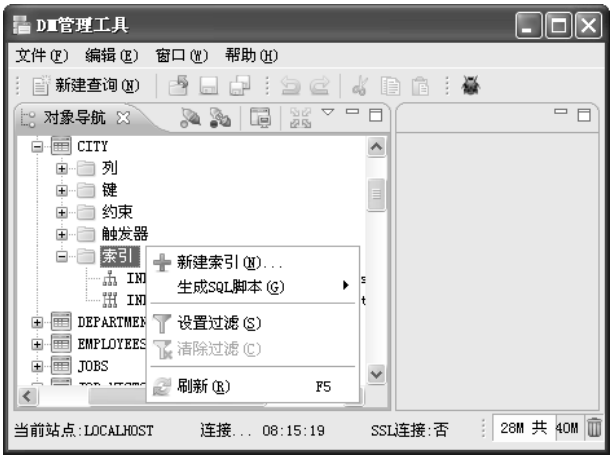

图 4-73　新建索引

图 4-74　设置位图连接索引常规参数

步骤 2：在图 4-74 中，索引名称设置为“REGION_NAME_IDX”，索引类型设置为“位图连接索引”，连接子句设置如下。

```
(REGION.REGION_NAME)
from DMHR.CITY, DMHR.REGION
where DMHR.CITY.REGION_ID=DMHR.REGION.REGION_ID
```

步骤 3：在图 4-74 中，单击“确定”按钮，完成位图连接索引的创建过程。

4.5.4　删除位图连接索引

删除位图连接索引与删除普通索引相同。

1. 用 SQL 命令删除位图连接索引

1）语法格式

删除位图连接索引的 SQL 命令格式如下：

```
DROP INDEX [<模式名>.]<索引名>;
```

2）应用举例

【例 4-67】以用户 SYSDBA 删除 SALES 模式下的 SALES_ CUSTOMER_NAME_IDX 位图连接索引。

```
SQL> DROP INDEX sales.sales_customer_name_idx;
```

2. 用管理工具删除位图连接索引

【例 4-68】以用户 SYSDBA 删除 DMHR 模式下 CITY 表的 REGION_NAME_IDX 位图连接索引。

步骤 1：在图 4-75 中右击 REGION_NAME_IDX 索引，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-76 所示。

图 4-75　删除位图索引

图 4-76　确认删除位图索引

步骤 2：在图 4-76 中，确认 REGION_NAME_IDX 无误后，单击“确定”按钮，删除位图索引。

4.5.5 创建全文索引

现有的数据库系统，绝大多数是以结构化数据为检索的主要目标，因此实现相对简单。例如，数值检索时，可以建立一张排序好的索引表，这样速度可以提高。但对于非结构化数据，即全文数据，要想实现检索，一般是采用模糊查询的方式实现的。这种方式不仅速度慢，还容易将汉字错误切分，于是产生了全文检索技术。

全文检索技术是智能信息管理的关键技术之一，其主要目的就是实现对大容量的非结构化数据的快速查找。DM 实现了全文检索功能，并将其作为 DM 服务器的一个较独立的组件，提供更加准确的全文检索功能，较好地解决了模糊查询方式带来的问题。

DM 全文检索根据已有词库建立全文索引，文本查询完全在索引上进行。全文索引为在字符串数据中进行复杂的词搜索提供了有效支持。用户可以在指定表的文本列上创建和删除全文索引。创建全文索引后全文索引未插入任何索引信息。当用户填充全文索引时，系统才将定义了全文索引的文本列的内容进行分词，并根据分词结果填充索引。用户可以在进行全文索引填充的列上使用 CONTAINS 谓词进行全文检索。

1. 用 SQL 命令创建全文索引

1）语法格式

创建全文索引的 SQL 命令格式如下：

```
CREATE CONTEXT INDEX <索引名> ON [<模式名>.] <表名> (<索引列定义>) [LEXER <分词参数>] [SYNC];
```

LEXER 关键词之后是分词参数，分词参数有 4 种，即 CHINESE_LEXER，中文最少分词；CHINESE_VGRAM_LEXER，中文最多分词；ENGLISH_LEXER，英文分词；DEFAULT_LEXER，中英文最少分词。其默认为 DEFAULT_LEXER。中文分词可以划分英文，但是指定英文分词不可以划分中文。

如果设置 SYNC 属性，则会同步填充索引信息；如果没有设置该属性，则需要用全文索引修改语句填充索引信息，之后才能进行全文检索。SYNC 属性只能保证创建全文索引时和表数据同步，如果之后表数据发生了改变，仍需要填充索引信息。在指定表的指定字段上建立全文索引完成后，全文索引信息会保存在 CTISYS 模式下的 SYSCONTEXTINDEXES 系统表中。

一条全文索引作用于表的一个文本字段，类型可为 CHAR、CHARACTER、VARCHAR、LONGVARCHAR、TEXT 或 CLOB，不允许为组合列和计算列，同一列只允许创建一个全文索引。

2）应用举例

【例 4-69】以用户 SYSDBA 给 PERSON 模式下的 ADDRESS 表的 ADDRES1 字段创建全文索引。

（1）创建全文索引：

```
SQL> CREATE CONTEXT INDEX idx_address ON person.address(address1) LEXER CHINESE_LEXER;
```

（2）使用模糊查询：

```
SQL> SELECT * FROM person.address WHERE address1 LIKE '%武昌%';
行号     ADDRESSID ADDRESS1              ADDRESS2 CITY      POSTALCODE PERSONID
---------- ---------- -------------------- -------- ---------- ---------- ----------
1          4         武昌区武船新村 115 号  武汉市武昌区          430063     10
2          11        武昌区武船新村 1 号    武汉市武昌区          430063     10
```

（3）使用全文索引查询：

```
SQL> SELECT * FROM person.address WHERE CONTAINS(address1, '武昌');
未选定行
```

在这个例子中，使用全文索引没有查询出应该有的数据，原因是全文索引创建成功后，并没有立即填充全文索引的内容，只有填充全文索引的内容后才能使用全文索引。

2. 用管理工具创建全文索引

【例 4-70】以用户 SYSDBA 给 DMHR 模式下 LOCATION 表的 STREET_ADDRESS 字段创建全文索引，采用中文最小分词。

步骤 1：在图 4-77 中右击“全文索引”节点，在弹出的快捷菜单中选择“新建全文索引”选项，弹出“新建全文索引”对话框，如图 4-78 所示。

步骤 2：在图 4-78 中，索引名称设置为 ADDRESS_CONT_IDX，模式名称选择 DMHR，表名选择 LOCATION，列名选择 STREET_ADDRESS（VARCHAR(50)），分词类型选择“中文最少分词”。单击“确定”按钮，完成创建全文索引过程。

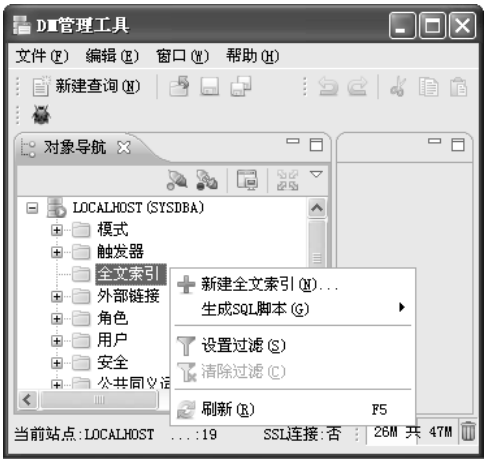

图 4-77　新建全文索引

图 4-78　设置全文索引参数

4.5.6　修改全文索引

全文索引的更新包括两种方式：完全更新和增量更新。

完全更新：删除原有的全文索引，对表进行全表扫描，逐一重构索引信息。在创建全文索引成功后，需完全更新全文索引才可以执行有效的全文检索。完全更新全文索引没有次数限制，用户根据需要在增量更新或者完全更新失败以及发生系统故障后都可以执行完全更新全文索引。

增量更新：在上一次完全更新之后，表数据发生了变化，此时可以使用增量更新来更新索引。增量更新只对新发生改变的数据进行重新分词生成索引信息，所以其代价是较小的。此时，也可以使用完全更新达到同样的更新索引的效果，但代价较大。

DM 在执行更新全文索引的过程中并没有产生相应的日志，即无法实现更新的回滚，如果在更新过程中发生任何错误，就必须执行完全更新获得正确的一致的全文索引信息。还需要注意的是当对已创建全文索引的表进行批量数据修改（如查询插入、批量导入等）时，需进行完全更新才能构建数据一致的新的全文索引，因为这些批量修改的数据信息未保存在全文索引动态信息表中，所以此时执行增量更新操作起不到更新全文索引的作用。

1. 用 SQL 命令修改全文索引

1）语法格式

修改全文索引的 SQL 命令格式如下：

```
ALTER CONTEXT INDEX <索引名> ON [<模式名>.] <表名> <REBUILD | INCREMENT>;
```

REBUILD 是完全填充全文索引，INCREMENT 是增量更新全文索引。创建全文索引后，需要通过修改全文索引的命令来填充全文索引信息。在完全填充全文索引后，如果表的数据再次发生变化，就需要增量式更新全文索引，之后全文检索才可以获得正确的结果。

2）应用举例

【例 4-71】以用户 SYSDBA 给 PERSON 模式下的 ADDRESS 表的 ADDRES1 列完全填

充全文索引。

（1）完全填充全文索引。

```
SQL> ALTER CONTEXT INDEX idx_address ON person.address REBUILD;
```

（2）模糊查询。

以“武昌”做模糊查询：

```
SQL> SELECT * FROM person.address WHERE address1 LIKE '%武昌%';
行号    ADDRESSID    ADDRESS1              ADDRESS2 CITY    POSTALCODE    PERSONID
---------- ---------- -------------------- -------- ---------- ---------- ----------
1           4          武昌区武船新村 115 号    武汉市武昌区          430063          10
2           11         武昌区武船新村 1 号      武汉市武昌区          430063          10
```

以“昌区”做模糊查询：

```
SQL> SELECT * FROM person.address WHERE address1 LIKE '%昌区%';
行号    ADDRESSID    ADDRESS1              ADDRESS2 CITY    POSTALCODE    PERSONID
---------- ---------- -------------------- -------- ---------- ---------- ----------
1           4          武昌区武船新村 115 号    武汉市武昌区          430063          10
2           11         武昌区武船新村 1 号      武汉市武昌区          430063          10
```

（3）全文检索。

以“武昌”做全文检索：

```
SQL> SELECT * FROM person.address WHERE CONTAINS(address1, '武昌');
行号    ADDRESSID    ADDRESS1              ADDRESS2 CITY    POSTALCODE    PERSONID
---------- ---------- -------------------- -------- ---------- ---------- ----------
1           4          武昌区武船新村 115 号    武汉市武昌区          430063          10
2           11         武昌区武船新村 1 号      武汉市武昌区          430063          10
```

以“昌区”做全文检索：

```
SQL> SELECT * FROM PERSON.ADDRESS WHERE CONTAINS(ADDRESS1, '昌区');
未选定行
```

这个例子说明，完全填充全文索引后，“武昌”被确定为全文检索词，能够通过全文检索查询到相关数据，“昌区”不是全文检索词，虽然用模糊查询可以查询到相关数据，但是用全文检索就查不到相关数据了。

2. 用管理工具修改全文索引

【例 4-72】以用户 SYSDBA 给 DMHR 模式下的 LOCATION 表的 STREET_ADDRES 字段完全填充全文索引。

在图 4-79 中右击 ADDRESS_CONT_IDX 全文索引，在弹出的快捷菜单中选择“完全填充”选项，完成对该索引的完全填充。

图 4-79　完成填充全文索引

4.5.7　删除全文索引

删除全文索引时，数据字典中相应的索引信息和全文索引内容都会被删除。删除索引有两种方式：一是使用语句删除，二是当模式对象发生改变时系统自动删除。第一种方式仅删除指定的表的全文索引信息；第二种方式会删除模式对象（如数据库、基表等）上的所有全文索引。全文索引一旦删除就不能回滚，基于该表的全文检索就会失败，只能通过重新构建获得新的全文索引。

1. 用 SQL 命令删除全文索引

1）语法格式

删除全文索引的 SQL 命令格式如下：

```
DROP CONTEXT INDEX <索引名> ON [<模式名>.] <表名>;
```

2）应用举例

【例 4-73】用户 SYSDBA 需要删除在 PERSON 模式下的 ADDRESS 表的 INDEX_ADDRESS 全文索引。

（1）删除全文索引：

```
SQL> DROP CONTEXT INDEX idx_address ON person.address;
```

（2）模糊查询：

```
SQL> SELECT * FROM person.address WHERE address1 like '%武昌%';
行号     ADDRESSID   ADDRESS1              ADDRESS2 CITY    POSTALCODE  PERSONID
---------- ----------- -------------------- -------- ------------ ---------- ----------
1          4           武昌区武船新村 115 号          武汉市武昌区     430063      10
2          11          武昌区武船新村 1 号            武汉市武昌区     430063      10
```

（3）尝试再次全文检索。

```
SQL> SELECT * FROM person.address WHERE CONTAINS(address1, '武昌');
SELECT * FROM person.address WHERE CONTAINS(address1, '武昌');
第 1 行附近出现错误[-3217]:列[ADDRESS1]未编制全文索引.
```

这个例子表明，删除全文索引后不会影响源表的数据，只是不能再使用全文检索了。

3）附加说明

除了该语句可删除全文索引外，当数据库模式发生如下改变时，系统将自动调用全文索引删除模块。

（1）删除表时，删除表上的全文索引。

（2）删除建立了全文索引的列时，删除列上的全文索引。

（3）修改建立了全文索引的列时，删除列上的全文索引。

2. 用管理工具删除全文索引

【例 4-74】以用户 SYSDBA 登录，删除 DMHR 模式下的 LOCATION 表的 ADDRESS_CONT_IDX 全文索引。

在图 4-79 中右击 ADDRESS_CONT_IDX 全文索引，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-80 所示。确认 ADDRESS_CONT_IDX 无误后，单击“确定”按钮，删除该全文索引。

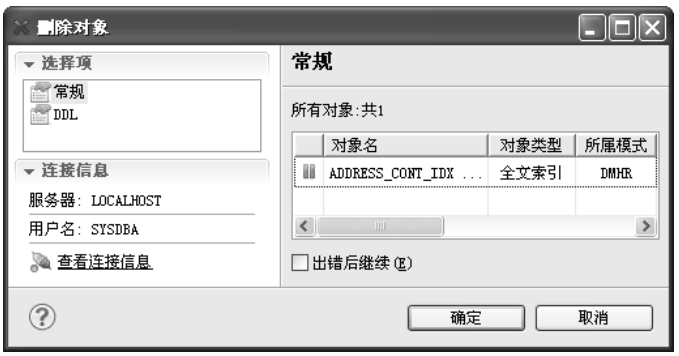

图 4-80　删除全文索引

4.6　序列管理

序列是达梦数据库中的数据库对象之一。通过使用序列，多个用户可以产生和使用一组不重复的有序整数值。例如，可以用序列来自动地生成主关键字值。序列通过提供唯一数值的顺序表来简化程序设计工作。当一个序列第一次被查询调用时，它将返回一个预定值，该预定值就是在创建序列时所指定的初始值。默认情况下，对于升序序列，序列的默认初始值为序列的最小值，对于降序序列，默认初始值为序列的最大值。可以指定序列能生成的最大值，默认情况下，降序序列的最大值为-1，升序序列的最大值为 $2^{31}-1$；也可以指定序列能生成的最小值，默认情况下，升序序列的最小值为 1，降序序列的最小值为 -2^{31}。

序列的最大值和最小值可以指定为 LONGINT（4 个字节）所能表示的最大和最小有符号整数。在随后的每一次查询中，序列将产生一个按其指定的增量增长的值。增量可以是任意的正整数或负整数，但不能为 0。如果此值为负，则序列是下降的；如果此值为正，则序列是上升的。默认情况下，增加为 1。

一旦序列生成，用户就可以在 SQL 语句中用以下伪列来存取序列的值。

（1）CURRVAL，返回当前的序列值。

（2）NEXTVAL，如果为升序序列，序列值增加并返回增加后的值；如果为降序序列，序列值减少并返回减少后的值。

序列可以是循环的，当序列的值达到最大值/最小值时，序列将重新从最小值/最大值计数。使用一个序列时，不保证生成一串连续不断递增的值。例如，如果查询一个序列的下一个值供 INSERT 使用，则该查询是能使用这个序列值的唯一会话。如果未能提交事务处理，则列值就不被插入表中，以后的 INSERT 将继续使用该序列随后的值。

序列在对编号的使用上具有很大用处，如果想对表建立一个字段专门用来表示编号，如订单号，这样就可以使用序列，依次递增生成，用户不需进行特殊管理，这给用户带来了很大方便。如果用户需要间隔的编号，创建序列时指定 INCREMENT，就可以生成用户需要的编号。

4.6.1 创建序列

下面用 SQL 命令和管理工具两种方式来创建序列。

1. 用 SQL 命令创建序列

1）语法格式

创建序列的 SQL 命令格式如下：

```
CREATE SEQUENCE [ <模式名>.] <序列名>  [ <序列选项列表>];
```

在<序列选项列表>中可以指定一种或多种序列选项，常见的参数说明见表 4-12。

表 4-12　参数说明

参　数	说　明
INCREMENT BY <增量值>	指定序列数之间的间隔，这个值可以是任意的 DM 正整数或负整数，但不能为 0。如果此值为负，序列是下降的，如果此值为正，序列是上升的。如果忽略 INCREMENT BY 子句，则间隔默认为 1
START WITH <初值>	指定被生成的第一个序列数，可以用这个选项来从比最小值大的一个值开始升序序列或比最大值小的一个值开始降序序列。对于升序序列，默认值为序列的最小值，对于降序序列，默认值为序列的最大值
MAXVALUE <最大值>	指定序列能生成的最大值，如果忽略 MAXVALUE 子句，则降序序列的最大值默认为-1，升序序列的最大值为 9223372036854775806（0x7FFFFFFFFFFFFFFE）。非循环序列在到达最大值之后，将不能继续生成序列数

（续表）

参　数	说　明
MINVALUE <最小值>	指定序列能生成的最小值，如果忽略 MINVALUE 子句，则升序序列的最小值默认为 1 ，降序序列的最小值为-9223372036854775808（0x8000000000000000）。循环序列在到达最小值之后，将不能继续生成序列数
CYCLE/NOCYCLE	CYCLE：该关键字指定序列为循环序列，当序列的值达到最大值/最小值时，序列将从最小值/最大值计数 NOCYCLE：该关键字指定序列为非循环序列，当序列的值达到最大值/最小值时，序列将不再产生新值
CACHE/NOCACHE	CACHE：该关键字表示序列的值是预先分配的，并保持在内存中，以便更快地访问；<缓存值>指定预先分配的值的个数，最小值为 2；最大值为 50000；且缓存值不能大于(<最大值> - <最小值>)/<增量值> NOCACHE：该关键字表示序列的值是不预先分配的
ORDER/NOORDER	ORDER：该关键字表示以保证请求顺序生成序列号 NOORDER：该关键字表示不保证请求顺序生成序列号

2）应用举例

【例 4-75】DMHR 模式下的 LOCATION 表已经存在 11 条记录，LOCATION_ID 的值分别为 1～11，现在要增加新记录，需要用序列值来填充 LOCATION_ID 的值。

（1）创建序列 SEQ_LOCID，初始值为 12，每次增加 1。

```
SQL> CREATE SEQUENCE dmhr.seq_locid START WITH 12 INCREMENT BY 1 ORDER;
```

（2）运用序列 SEQ_LOCID，给 LOCATION 表增加两条记录。

```
SQL> INSERT INTO dmhr.location(location_id, street_address, postal_code, city_id)
VALUES(dmhr.seq_locid.NEXTVAL, '江岸区香港路 8 号', '430010', 'WH');
SQL> INSERT INTO dmhr.location(location_id, street_address, postal_code, city_id)
VALUES(dmhr.seq_locid.NEXTVAL, '雁塔区太白南路 2 号', '710071', 'XA');
```

（3）查询 LOCATION 表数据，检验序列使用效果。

```
SQL> SELECT * FROM dmhr.location;
```

行号	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY_ID
1	1	海淀区北三环西路 48 号	100086	BJ
2	2	桥西区槐安东路 28 号	050000	SJ
3	3	浦东区张江高科技园博霞路 50 号	201203	SH
4	4	江宁开发区迎翠路 7 号	210000	NJ
5	5	天河区体育东路 122 号	510000	GZ
6	6	龙华区玉沙路 16 号	570100	HK
7	7	东湖开发区关山一路特 1 号	430074	WH
8	8	天心区天心路 4 号	410000	CS
9	9	沈河区沈阳路 171 号	110000	SY
10	10	雁塔区雁塔南路 10 号	710000	XA
11	11	金牛区人民北路 1 号	610000	CD

12	12	江岸区香港路 8 号	430010	WH
13	13	雁塔区太白南路 2 号	710071	XA

13 rows got

第 12 条记录的 LOCATION_ID 值为 12，表明使用了 SEQ_LOCID 序列的初始值；第 13 条记录 LOCATION_ID 的值为 13，表明 SEQ_LOCID 序列每次增加 1。

2. 用管理工具创建序列

【例 4-76】PURCHASING 模式下的 VENDOR 表有 12 条记录，VENDORID 字段的值为 1～12，现在创建一个序列 SEQ_VENDORID，当给 VENDOR 表增加记录时，用 SEQ_VENDORID 序列的值来填充 VENDORID 字段。

步骤 1：在图 4-81 中右击 PURCHSING 模式下的“序列”节点，在弹出的快捷菜单中选择“新建序列”选项，弹出“新建序列”对话框，如图 4-82 所示。

图 4-81　新建序列

图 4-82　设置序列参数

步骤 2：在图 4-82 中，序列名设置为 SEQ_VENDORID，起始值设置为 13，增量设置为 1。单击“确定”按钮，完成创建序列过程。

4.6.2　删除序列

序列与其他数据库对象没有直接关系和依赖关系，删除序列对其他数据库对象没有影响。如果一个序列生成器当前值为 150，用户想要从值 27 开始重新启动此序列生成器，则可以先删除此序列生成器，然后重新以相同的名称创建序列生成器，START WITH 选项值为 27。

1. 用 SQL 命令删除序列

1）语法格式

删除序列的 SQL 命令格式如下：

```
DROP SEQUENCE [ <模式名>.]<序列名>;
```

2）应用举例

【例 4-77】更改序列生成器的值。DMHR 模式下的序列 SEQ_LOCID 当前值为 50，希望从 15 开始重新启用此序列生成器。

（1）删除序列：

```
SQL> DROP SEQUENCE dmhr.seq_locid;
```

（2）创建同名序列：

```
SQL> CREATE SEQUENCE dmhr.seq_locid START WITH 15 INCREMENT BY 1 ORDER;
```

2. 用管理工具删除序列

【例 4-78】以用户 SYSDBA 删除 PURCHASING 模式下的序列 SEQ_VENDORID。

步骤 1：在图 4-83 中右击 PURCHASING 模式下的序列 SEQ_VENDORID，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-84 所示。

图 4-83　删除序列

图 4-84　确认删除序列

步骤 2：在图 4-84 中，确认 SEQ_VENDORID 无误后，单击“确定”按钮，删除该序列。

4.7　同义词管理

同义词让用户能够为数据库的一个模式下的对象提供别名。同义词通过掩盖一个对象真实的名字和拥有者，并且对远程分布式的数据库对象给予位置透明特性以此来提供一定的安全性，并使用同义词简化复杂的 SQL 语句。同义词可以替换模式下的表、视图、序列、函数、存储过程等对象。

同义词相当于模式对象的别名，起着连接数据库模式对象和应用程序的作用。假如模式对象需要更换或者修改，则不用修改应用程序而直接修改同义词即可。

同义词是用来实现下列用途的数据库对象。

（1）为本地或远程服务器上的其他数据库对象（称为基础对象）提供备用名称。

（2）提供抽象层，以免客户端应用程序对数据库对象的名称或位置进行更改。

同义词的好处在于用户可能需要某些对象在不同的场合采用不用的名称，使其适合不同人群的应用环境。例如，创建表 product，如果客户不认识这个英文词，则可以增加同义词，命名为“产品”，这样客户就有了较直观的观念，一目了然。

4.7.1　创建同义词

1. 用 SQL 命令创建同义词

1）语法格式

创建同义词的 SQL 命令格式如下：

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [<模式名>.]<同义词名>  FOR [<模式名>.]<对象名>
```

FOR 关键字之后的模式名、对象名就是与同义词等同的对象。

2）应用举例

【例 4-79】以用户 SYSDBA 创建 PURCHSING 模式的 VENDOR 表的全局同义词 SYNG_VENDOR 和非全局同义词 SYNN_VENDOR，然后以 DMHR 用户登录，分别查询

SYNG_VENDOR 和 SYNN_VENDOR。

（1）创建 VENDOR 表的全局同义词 SYNG_VENDOR：

```
SQL> CREATE PUBLIC SYNONYM syng_vendor FOR purchasing.vendor;
```

（2）创建 VENDOR 表的非全局同义词 SYNN_VENDOR：

```
SQL> CREATE SYNONYM dmhr.synn_vendor FOR purchasing.vendor;
```

（3）给 DMHR 用户授予查询 VENDOR 表的权限：

```
SQL> GRANT SELECT ON purchasing.vendor TO dmhr;
```

（4）以 DMNR 用户登录数据：

```
SQL> CONN dmhr/DMHR12345;
```

（5）查询全局同义词 SYNG_VENDOR 表：

```
SQL> SELECT * FROM syng_vendor WHERE vendorid>10;
```

行号	VENDORID	ACCOUNTNO	NAME	ACTIVEFLAG	WEBURL	CREDIT
1	11	00	机械工业出版社	1		1
2	12	00	文学出版社	1		1
3	13	00	北京大学出版社	1		1

（6）查询非全局同义词 SYNN_VENDOR：

```
SQL> SELECT * FROM synn_vendor WHERE vendorid>10;
```

行号	VENDORID	ACCOUNTNO	NAME	ACTIVEFLAG	WEBURL	CREDIT
1	11	00	机械工业出版社	1		1
2	12	00	文学出版社	1		1
3	13	00	北京大学出版社	1		1

这个例子说明，只有对原对象有操作权限，才能对同义词有操作权限。对同义词的操作等同于对原对象的操作。

3）附加说明

（1）全局同义词创建时不能指定同义词的模式名限定词，它能够被所有用户使用，使用时不需要加任何模式限定名。非全局同义词被其他用户引用需要在前面加上模式名；公有同义词和私有同义词可以具有相同的名称。

（2）同义词创建时，并不会检查其所指代的同义词对象是否存在，用户使用该同义词时，如果不存在指代对象或者对该指代对象不拥有权限，则会报错。

（3）用户使用 SQL 语句对某个对象进行操作，那么解析一个对象的顺序，首先是查看模式内是否存在该对象，再查看模式内的同义词（非全局同义词），最后才是全局同义词。

例如，用户 OE 和 SH 在其模式下都有一个表称为 customer，SYSDBA 为 OE 模式下的 customer 表创建了一个全局同义词 customer_syn，SYSDBA 为 SH 模式下的 customer 表创建了一个私有同义词 customer_syn，如果用户 SH 查询，即 SELECT COUNT(*) FROM customer_syn，则此时返回的结果为 SH.CUSTOMER 下的行数，而如果需要访问 OE 模式下的 CUSTOMER 表，则必须在前面加模式名（此时全局同义词 CUSTOMER 失效）：

SELECT COUNT(*) FROM OE.customer_syn;。

2. 用管理工具创建同义词

【例 4-80】以用户 SYSDBA 创建 DMHR 模式的 LOCATION 表的非全局同义词 SYN_LOCATION，查询 SYN_LOCATION，检验同义词的使用。

步骤 1：在图 4-85 中右击 DMHR 模式下的“同义词”节点，在弹出的快捷菜单中选择“新建同义词”选项，弹出“新建同义词”对话框，如图 4-86 所示。

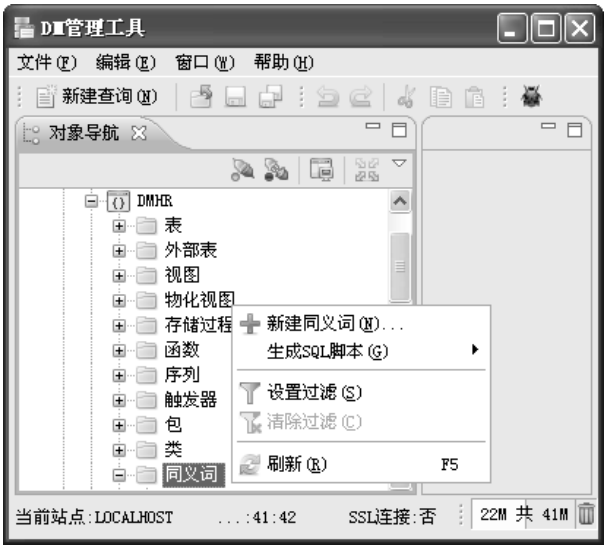

图 4-85　新建同义词

图 4-86　设置同义词参数

步骤 2：在图 4-86 中，设置同义词名为 SYN_LOCATION，对象所属模式为 DMHR，对象名为 LOCATION。单击“确定”按钮，完成新建同义词的过程。

4.7.2　删除同义词

同义词只是数据库对象的一个别名，删除同义词不会删除原数据库对象。删除公有同义词，需要指定 PUBLIC，而删除私有同义词不能指定，否则报错；如果删除当前模式下的同义词，则可以不指定模式名，如果删除其他模式下的同义词，则需要指定相应的模式名，否则报错。

1. 用 SQL 命令删除同义词

1）语法格式

删除同义词的 SQL 命令格式如下：

```
DROP [PUBLIC] SYNONYM <同义词名>
```

2）应用举例

【例 4-81】以用户 SYSDBA 删除全局同义词 SYNG_VENDOR。

```
SQL> CONN SYSDBA/SYSDBA;
SQL> DROP PUBLIC SYNONYM syng_vendor;
```

【例 4-82】以用户 SYSDBA 删除 DMHR 模式下的同义词 SYNN_VENDOR。

（1）删除同义词：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> DROP SYNONYM dmhr.synn_vendor;
```

（2）继续查询原表数据：

```
SQL> SELECT * FROM PURCHASING.VENDOR WHERE VENDORID>10;
行号     VENDORID    ACCOUNTNO    NAME          ACTIVEFLAG WEBURL CREDIT
---------- ----------- --------- -------------- ---------- ------ ----------------------------------------------------------------
1          11          00        机械工业出版社 1                  1
2          12          00        文学出版社     1                  1
3          13          00        北京大学出版社 1                  1
```

这个例子表明，删除数据库对象的同义词不会对原数据库对象产生影响。

2. 用管理工具删除同义词

【例 4-83】以用户 SYSDBA 删除 DMHR 模式下的同义词 SYN_LOCATION。

步骤 1：在图 4-87 中右击 DMHR 模式下的同义词 SYN_LOCATION，在弹出的快捷菜单中选择“删除”选项，弹出“删除对象”对话框，如图 4-88 所示。

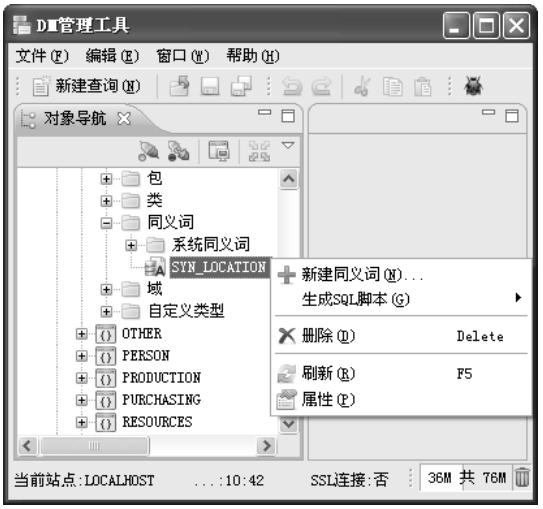

图 4-87　删除同义词

图 4-88　确认删除同义词

步骤 2：在图 4-88 中，确认 SYN_LOCATION 无误后，单击“确定”按钮，完成同义词的删除过程。

第 5 章

备份与还原

数据库的备份与还原是系统容灾的重要方法。在一个生产系统中，数据库往往处于核心的地位，为了保证数据的安全，人们想出了各种各样的方法，备份与还原是其中一种重要的方法。备份意味着把重要的数据复制到安全的存储介质上，还原则意味着在必要的时候再把以前备份的数据复制到最初的位置，以保证用户可以访问这样的数据。为了提高备份还原的安全性，减少备份文件占用磁盘空间的大小，DM 支持对备份数据的加密和压缩。

5.1 备份还原概述

5.1.1 相关概念

1. 备份

备份就是将待备份的数据经过处理（如加密、压缩等）后，写到备份片文件中，并将相关备份信息写到元数据文件中的过程。备份的目的是，当数据库遇到损坏的情况时，可以执行还原恢复操作，把数据库复原到损坏前的某个时间点。

2. 备份集

备份集用来存放备份过程中产生的备份数据及备份信息。一个备份集对应了一次完整的备份。一个备份集为一个目录，它由一个或多个备份片文件和一个元数据文件组成。

3. 备份片

备份片是用来存储备份数据的文件。备份时，数据文件内容或归档日志内容经过处理后，都会存放到这些备份片文件中。备份片文件后缀为.bak。

4. 备份元数据

元数据文件用来存储备份信息，通过元数据文件，可以了解整个备份集信息。元数据文件后缀为.meta。

5. 还原

还原是备份的逆过程，就是把备份集中的备份数据经过处理后，写回到还原目标库中相应的数据文件中的过程。

6. 恢复

恢复是重做本地归档日志或者备份集中备份的归档日志的过程。利用恢复操作，可以使数据库恢复到备份时，或者某个最新状态。没有经过恢复的还原数据库是不允许启动的，因为还原回来的数据通常处于非一致性状态，需要执行恢复操作，使得目标数据库数据一致，才能对外提供服务。表空间还原与恢复是合并在一次操作中完成的。

7. 备份库

备份库指需要进行备份的库，又称源库或源备份库。

8. 还原库

还原库指用来做还原的库，又称目标库或还原目标库。

9. 恢复库

恢复库指已经做过恢复的库。

5.1.2 备份还原分类

1. 备份分类

1）逻辑备份与物理备份

物理备份，指根据备份范围（数据库级、表空间级、表级）将数据文件中有效数据页和归档日志（也可能没有归档日志，这需要用户来指定）复制到备份片文件中的过程。这种备份是在文件层进行的。

逻辑备份，指利用 DM 7 提供的逻辑导出工具 DEXP，将指定对象（数据库级、模式级、表级）的数据导出到文件的备份。

在这两种方式中，物理备份是更强健的数据保护方式，也是备份策略中的首选。逻辑备份是物理备份的补充方式，相对物理备份而言，具有更大的灵活性。

2）联机备份与脱机备份

按照数据库的状态，可以把备份划分为联机备份和脱机备份。

联机备份，指数据库处于运行状态，通过执行 SQL 语句进行的备份。当前许多系统都要求 7×24 小时提供服务，联机备份是最常用的备份形式之一。联机备份时，大量的事务处于活动状态，为确保备份数据的一致性，需要同时备份一段日志（备份期间产生的 REDO 日志）。按照联机备份要求，数据库必须配置本地归档，且归档必须处于开启状态。

脱机备份，指数据库处于关闭状态时，使用 DMRMAN 工具执行的备份。需要注意的是，只有正常关闭的数据库才允许执行脱机备份。正在运行或异常关闭的数据库无法成功执行脱机备份，系统会报错。

3）库备份、表空间备份与表备份

按照备份的粒度大小，可以将备份划分为数据库备份、表空间备份和表备份。

库备份，指对整个数据库执行的备份，又称为库级备份。库备份的对象是数据库中所有数据文件和备份过程中的归档日志，可选择是否备份日志。

表空间备份，指对表空间执行的备份，又称为表空间级备份。表空间备份的过程就是复制表空间内所有数据文件的有效数据的过程。DM 7 不允许对 SYSTEM、ROLL、TEMP 表空间进行备份还原。

表备份，指将表的所有数据页备份到备份集中，并记录各个数据页之间的逻辑关系用来恢复表数据结构。表备份不需要备份归档日志，不存在增量备份之说。DM 7 仅支持单个用户表备份或者分区表的单个子分区表的备份。

4）一致性备份与非一致性备份

一致性备份，指备份集中包含了全部的备份数据。可以仅利用备份集中的备份数据就把数据库恢复到备份时的状态，如联机库备份（带日志）、脱机库备份等。

非一致性备份，指单独使用备份集中的数据还不足以把数据库还原到备份时某个数据一致性的点，需要借助归档来恢复。

5）完全备份与增量备份

完全备份，指备份中包含了指定的库（或者表空间）的全部数据页，这样的一个备份通常会很大，且备份持续时间也比较长。对于一个需要经常备份的系统，执行完全备份是比较消耗时间和空间的。

增量备份，指基于某个已有的备份（完全备份或者增量备份），备份自该备份以来所有发生修改了的数据页。这个已有的备份称为基备份。相对而言，增量备份通常很小，备份也较快且占用空间也会比较少。

由于增量备份是基于某个已有备份集进行的备份，这样的依赖关系就构成了一个备份集链表。一个完整的备份集链表，必须包含一个完全备份，且这个完全备份一定是链表中的第一个备份。若备份集链表中存在多个备份集，则其他位置的备份集均为增量备份集，且越往后备份集越新，最后一个备份集为最新生成的备份集。若备份集链表中只有一个备份集，那么这个备份集一定是完全备份。

2. 还原分类

1）逻辑还原与物理还原

逻辑还原，是逻辑备份的反过程，指使用 DM 7 提供的 DIMP 工具把使用 DEXP 导出的备份数据重新导入的过程。

物理还原，是物理备份的逆过程，可以通过联机执行 SQL 语句，或者通过 DMRMAN 等脱机工具，把备份时得到的备份集还原到目标数据文件的过程。

2）联机还原与脱机还原

联机还原，指数据库处于运行状态时，通过执行 SQL 语句完成的还原过程。

脱机还原，指数据库处于脱机状态时，通过 DMRMAN 工具进行的还原过程。还原的目标库必须是新初始化或者处于正常关闭状态的数据库。

3）数据库还原、表空间还原与表还原

按照备份粒度大小，还原分为数据库还原、表空间还原和表还原。

可以将源库作为还原目标库，但若还原过程失败，则目标库将被损坏，不能使用，因此建议不要在源库上进行库还原。DM 7 支持从库备份集中还原指定的表空间，也允许从库备份级和表空间备份集中还原指定的数据文件。表还原实质上是表内数据的还原，以及索引和约束等的重建。

4）完全备份还原与增量备份还原

根据备份集，将还原分为完全备份还原和增量备份还原。

完全备份还原，指目标还原备份集为完全备份。完全备份还原可以不依赖其他备份集直接完成还原操作。

增量备份还原，指目标还原备份集为增量备份。增量备份还原需要完整的备份集链表才能完成还原操作。因此，增量备份还原时需要用户确保完整备份集链表中各备份集都存在，否则将无法执行。

3. 恢复分类

1）更新 DB_MAGIC

首先介绍一下 DM 7 的 permenant_magic 和 DB_MAGIC，两者都是 DM 7 在初始化数据库时自动生成的用来标识数据库唯一性的值。permenant_magic 一经生成，永久不变，称为数据库永久魔数。每一个库有且仅有一个数据库永久魔数。DB_MAGIC 则记录着数据库的变化，如数据库经过备份还原后，DB_MAGIC 就会改变。

执行还原后，如果最后的备份集在备份过程中无日志生成（如脱机备份），那么此时还原数据库中的数据与备份时的数据一致，不需要重做归档日志，可以通过直接更新 DB_MAGIC 来完成最后的恢复工作，否则还原目标库将无法启动。

2）从备份集恢复

如果备份过程中生成了日志，且这些日志在备份集中有完整备份，在还原后，可以重做备份集中备份的日志，将数据库恢复到备份时的状态。这个利用备份集中备份日志的恢复过程，可以看做从备份集恢复。

3）从归档恢复

利用本地归档日志来恢复数据的过程，称为从归档恢复。从归档恢复可以恢复到指定的时间点及指定的 LSN 值。若同时指定了时间点和 LSN，则以较早的为结束点。

（1）恢复到指定时间点：DM 7 中事务提交时，系统会生成一个特殊的 REDO 日志，记录事务提交的时间。重做归档日志时，一旦碰到比指定时间点更大的事务提交时间 REDO 日志，马上终止重做归档日志过程。用户可以通过指定一个时间点，使数据库恢复到这个指定的时间点。例如，用户在下午 5 点做了一个误操作，删除了某些重要数据；此时可以指定恢复时间点到下午 4:59，恢复被误删除的数据。

（2）恢复到指定 LSN：DM 7 中每条 REDO 日志记录都有一个 LSN 值，用户可以指定一个 LSN 值，将数据库恢复到产生指定 LSN 值时间点的状态。

（3）备份集恢复：执行备份集还原后，若不使用备份集恢复或者备份集为 WITHOUT LOG（不备份归档日志）的联机备份，则需要使用归档恢复，并指定 UNTIL LSN 不能小于备份集的 END LSN。

（4）恢复到最新：若不指定恢复到的时间点和恢复到的 LSN，则会重做所有本地归档，将数据库恢复到尽可能新的状态。

5.1.3　备份还原条件

1. 数据库备份还原条件

（1）数据库备份条件：联机备份时，数据库必须配置本地归档，且归档必须处于开启状态。

脱机备份时，只有正常关闭的数据库才允许脱机备份。

（2）数据库还原条件：数据库必须处于脱机状态。

2. 表空间备份还原条件

（1）表空间备份条件：不允许备份 SYSTEM 表空间、ROLL 表空间和 TEMP 表空间。

（2）表空间还原条件：数据库必须处于联机状态。

表空间还原本身包含恢复操作，因此还原后不需要再执行恢复操作。

3. 表备份还原条件

（1）表备份条件：数据库必须处于联机状态。

只能进行完全备份，不需要备份归档日志。

（2）表还原条件：数据库必须处于联机状态。

表还原本身包含恢复操作，因此还原后不需要再执行恢复操作。

5.2 数据库备份还原

5.2.1 使用 SQL 语句备份

1. 语法格式

```
BACKUP DATABASE [FULL | INCREMENT WITH BACKUPDIR '<备份目录>'{,'<备份目录>'} [USE PWR]][TO <备份名>]  BACKUPSET '<备份集路径>' [DEVICE TYPE <介质类型> [PARMS '<介质参数>']] [BACKUPINFO '<备份描述>'] [MAXPIECESIZE <备份片限制大小>] [IDENTIFIED BY <密钥>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]][COMPRESSED [LEVEL <压缩级别>]] [WITHOUT LOG][TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>] [PARALLEL [<并行数>]];
```

各参数说明如下：

（1）FULL|INCREMENT：备份类型，FULL 表示完全备份，INCREMENT 表示增量备份。

（2）<备份目录>：用于增量备份中指定备份目录，最大长度为 256 个字节；备份目录的搜索范围是，指定目录的当前目录及其第一级子目录，且优先判断当前目录是否存在.meta 文件，若是则停止搜索。例如，D:\tpcc\bak2\bak2_1\bak2_2，对于 D:\tpcc\bak2 来说，bak2_1 是其第一级子目录，bak2_2 是第二级子目录。若指定搜索目录为 D:\tpcc\bak2，则首先判断 D:\tpcc\bak2 中是否存在.meta，若存在，则停止搜索；否则，进入 bak2_1 判断是否存在.meta 文件，若不存在，则直接报错返回，不会进入 bak2_2。在实际应用中，备份目录为默认备份目录和当前备份执行备份集目录的上级目录。

（3）USE PWR：用于增量备份中，指定备份过程中使用 PWR 优化，默认不使用。

（4）备份名：指定生成备份名称。若未指定，系统随机生成，默认备份名格式为 DB_备份类型_数据库名_备份时间。

（5）备份集路径：指定当前备份集生成路径，若未指定，则在默认备份路径中生成备份集路径。

（6）介质类型：指存储备份集的设备类型，暂支持 DISK 和 TAPE，默认 DISK。

（7）介质参数：只对介质类型为 TAPE 时有效。

（8）备份描述：备份的描述信息。

（9）备份片限制大小：最大备份片文件大小上限，以 MB 为单位，最小 128MB，32 位系统最大 2GB，64 位系统最大 128GB。

（10）密钥：备份加密通过使用 IDENTIFIED BY 来指定加密密码。密码应用双引号括起来，这样避免了一些特殊字符通不过语法检测的问题。

（11）WITH ENCRYPTION <TYPE> ：指定加密类型，0 表示不加密，1 表示简单加密，2 表示复杂加密，若指定密钥或者加密算法，但若未指定加密类型，则默认加密类型为 1。

（12）加密算法：加密算法具体见《DM_SQL》，默认情况下，算法为 AES256_CFB。

（13）压缩级别：取值为 0～9。0 表示不压缩，1 表示 1 级压缩，9 表示 9 级压缩。压缩级别越高，压缩越慢，但压缩比越高。若未指定，但指定了 COMPRESSED，则默认为 1；否则，默认为 0。

（14）WITHOUT LOG：联机数据库备份是否备份日志。如果使用，则表示不备份，否则表示备份。如果使用了 WITHOUT LOG 参数，则使用 DMRMAN 工具还原时，必须指定 ARCHIVE_DIR 参数。

（15）trace 文件名：指定生成的 trace 文件。启用 trace，但不指定 trace file 时，默认在 DM 数据库系统的 log 目录下生成 DM_SBTTRACE_年月.log 文件；若使用相对路径，则生成在执行码同级目录下。

（16）trace 日志级别：有效值 1、2，默认为 1，表示不启用 trace，此时若指定了 trace file，会生成 trace 文件，但不写入 trace 信息；为 2 则启用 trace 并写入 trace 相关内容。

（17）并行数：指定并行备份的并行数，若不指定，则默认为 4，指定 0 或者 1 均认为非并行备份。增量备份中强制并行数与基备份一致。若未指定关键字 PARALLEL，则认为非并行备份。并行备份不支持 PWR 和存在介质为 TAPE 的备份。

2. 举例说明

【例 5-1】对数据库 DAMENG 进行联机完全备份，备份名为 DAMENG_BAK1。

（1）配置并启动归档模式：

```
SQL>CONN SYSDBA/SYSDBA;
SQL>ALTER DATABASE MOUNT;
SQL>ALTER DATABASE NOARCHIVELOG;
SQL>ALTER DATABASE ADD ARCHIVELOG 'DEST=C:\LOG, TYPE=LOCAL, FILE_SIZE=64,
SPACE_LIMIT=0';
SQL>ALTER DATABASE ARCHIVELOG;
SQL>ALTER DATABASE OPEN;
```

（2）重启数据库服务：

在 MD 服务查看器中，选择达梦数据库实例服务，单击“重启动”此服务的链接，重启数据库服务。

（3）完全备份：

```
SQL>BACKUP DATABASE FULL TO DAMENG_BAK1 BACKUPSET
'c:\dmdbms\data\DAMENG\bak\DMBAK1' BACKUPINFO 'DAMENG 的联机全库备份';
```

这个例子说明，DM 7 的联机库级和表空间级备份要求必须配置本地归档并启动归档模式，如果是非归档模式，则要进行配置。执行上述联机完全备份后，在 c:\dmdbms\data\DAMENG\bak\DMBAK1 目录下生成 DMBAK1.bak、DMBAK1.meta、DMBAK1_1.bak 等备份片文件和备份元数据文件。

【例 5-2】利用例 5-1 中的基备份目录对数据库 DAMENG 进行联机增量备份，备份名为 DAMENG_BAK1_INC。

```
SQL>BACKUP DATABASE INCREMENT WITH BACKUPDIR 'c:\dmdbms\data\
DAMENG\bak\DMBAK1' TO DAMENG_BAK1_INC BACKUPSET 'c:\dmdbms\data\
DAMENG\bak\DMBAK2' BACKUPINFO 'DAMENG 的联机增量备份';
```

执行上述联机增量备份后，在 c:\dmdbms\data\DAMENG\bak\DMBAK2 目录下生成 DMBAK2.bak、DMBAK2.meta 等备份片文件和备份元数据文件。

5.2.2 使用 DMRMAN 备份还原

DMRMAN 是 DM 7 的脱机备份还原管理工具，由它来统一负责库级脱机备份、脱机还原、数据库恢复等相关操作，该工具支持命令行指定参数方式和控制台交互方式执行，降低了用户的操作难度。

DMRMAN 工具支持命令行指定语句、命令行指定脚本和控制台输入三种操作方式，这三种方式下均允许执行上述功能所对应的操作语句。

DMRMAN 命令中的 CTLFILE 参数是可选的，指定执行语句所在的文件路径。脚本文件格式支持*.txt。CTLSTMT 参数是可选的，指定待执行语句，如 CTLSTMT="BACKUP DATABASE ‘D:\DMDBMS\DAMENG\dm.ini’"。HELP 参数是可选的，显示帮助信息。

1. 语法格式

```
BACKUP DATABASE '<INI 文件路径>' [FULL |INCREMENT [WITH BACKUPDIR ‘<基备份搜索
目录>’{,‘<基备份搜索目录>’}] [BASE ON BACKUPSET '<基备份集目录'>] [USE PWR]] [TO <备份名
>] [BACKUPSET ‘<备份集目录>’][DEVICE TYPE <介质类型>[PARMS ‘<介质参数>’] [BACKUPINFO
‘<备份描述>’] [MAXPIECESIZE <备份片限制大小>] [IDENTIFIED BY <密钥>[WITH
ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]][COMPRESSED [LEVEL < 压缩级别>]]
[PARALLEL [<并行数>]];
```

各参数说明如下：

（1）INI 文件路径：待备份目标数据库 dm.ini 文件路径。

（2）FULL|INCREMENT：备份类型，FULL 表示完全备份，INCREMENT 表示增量备份。

（3）基备份集目录：用于增量备份，为增量备份指定基备份集目录。

（4）备份名：指定生成备份名称。若未指定，系统随机生成，默认备份名格式为 DB_备份类型_数据库名_备份时间。

（5）备份集目录：指定当前备份集生成目录，若未指定，则在默认备份目录中生成备份集目录。

2. 应用举例

【例 5-3】脱机备份数据库。

（1）数据库脱机。

使用 DMRMAN 备份属于脱机备份，因此在备份数据库之前需要数据库脱机。运行达

梦服务查看器，选择 DM 数据库服务实例，单击停止链接即可使数据库脱机。

（2）备份数据库。

```
C:\dmdbms\bin>DMRMAN.exe
RMAN>BACKUP DATABASE 'C:\dmdbms\data\DAMENG\dm.ini'
```

命令执行完成后，在 C:\dmdbms\data\DAMENG\bak 目录下创建了一个 DB_DAMENG_20160728_000629_000950，该目录下有 DB_DAMENG_ 20160728_000629_ 000950.bak 数据片文件和 DB_DAMENG_20160728_ 000629_000950.meta 元数据文件。

5.2.3　使用 DMRMAN 还原恢复

1. 语法格式

```
RESTORE DATABASE '<INI 文件路径>' FROM BACKUPSET '<备份集目录>' [DEVICE TYPE DISK|TAPE[PARMS '<介质参数>']][IDENTIFIED BY <密钥> [ENCRYPT WITH <加密算法>]] [WITH BACKUPDIR '<基备份集搜索目录>'{,'<基备份集搜索目录>'}][MAPPED FILE '<映射文件>'][TASK THREAD <任务线程数>] [NOT PARALLEL];
```

主要参数说明如下：

（1）INI 文件路径：目标还原库 dm.ini 文件路径。

（2）备份集目录：指定待还原备份集目录。

2. 应用举例

使用 RESTORE 语句完成脱机还原操作，在还原语句中指定库级备份集，可以是脱机库级备份集，或联机库级备份集。

【例 5-4】还原数据库。将 DB_DAMENG_20160728_000629_000950 库备份集还原到 DM SERVERRES 数据库中。

（1）初始化 DM SERVERRES 数据库：

```
C:\dmdbms\bin>DMINIT PATH=c:\dmdbms\data\DM SERVERRES
```

初始化命令执行完毕后，会在 c:\dmdbms\data\DM SERVERRES\DAMENG 目录下创建系列文件和目录。

（2）还原数据库：

```
RMAN>RESTORE DATABASE 'C:\dmdbms\data\DM SERVERRES\DAMENG\dm.ini' FROM BACKUPSET 'C:\dmdbms\data\DAMENG\bak\DB_DAMENG_20160728_000629_ 000950'
```

（3）恢复数据库：

```
RMAN>RECOVER DATABASE 'C:\dmdbms\data\DM SERVERRES\DAMENG\dm.ini' FROM BACKUPSET 'C:\dmdbms\data\DAMENG\bak\DB_DAMENG_20160728_000629_ 000950'
```

（4）配置数据库服务：运行达梦数据库配置助手，如图 5-1 所示，选中“注册数据库服务”单选按钮，单击“开始”按钮，如图 5-2 所示，配置数据库的 INI 文件，指定实例的名称为 DMSEVERRES，端口号为 5256（不与其他服务冲突），单击“完成”按钮，进行服务

注册和启动，如图 5-3 所示。服务启动后，显示数据库服务的基本信息，如图 5-4 所示。

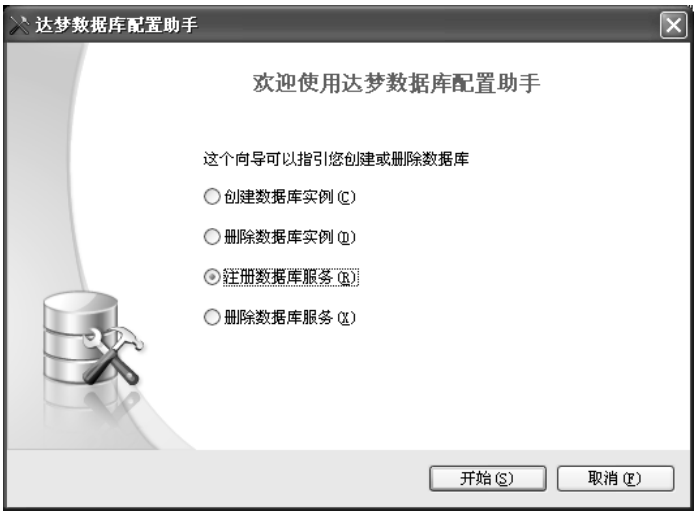

图 5-1　注册数据库服务

图 5-2　配置 INI 文件、实例名和端口号

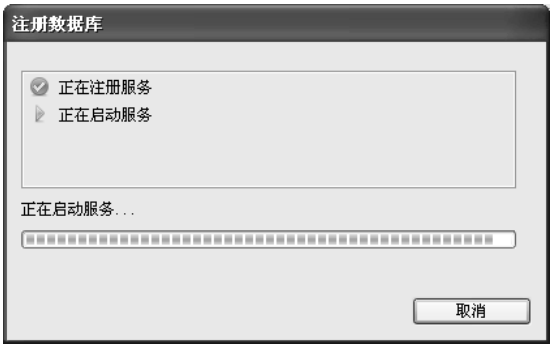

图 5-3　正在注册服务和启动服务

图 5-4　显示数据库服务的基本信息

5.3　表空间备份还原

5.3.1　使用 SQL 语句备份

1. 语法格式

BACKUP TABLESPACE <表空间名> [FULL | INCREMENT　WITH BACKUPDIR　‘<备份目录>’ {,‘<备份目录>’}[BASE ON　<BACKUPSET　‘<基备份目录>’] [USE PWR]] [TO <备份名>] BACKUPSET　‘<备份集路径>’[DEVICE TYPE <介质类型> [PARMS '<介质参数>']] [BACKUPINFO　‘<备份集描述>’] [MAXPIECESIZE <备份片限制大小>] [IDENTIFIED BY <密钥>[WITH ENCRYPTION<TYPE>] [ENCRYPT WITH <加密算法>]][COMPRESSED [LEVEL <压缩级别>]] [TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>] [PARALLEL [<并行数>]];

主要参数说明如下：

（1）表空间名：需要备份的表空间的名称，只能备份用户表空间。

（2）FULL|INCREMENT：备份类型，FULL 表示完全备份，INCREMENT 表示增量备份。

（3）基备份目录：用于增量备份，为增量备份指定基备份目录。

（4）备份名：指定生成备份名称。若未指定，则系统随机生成，默认备份名格式为 TS_备份类型_表空间名_备份时间。

（5）备份集路径：指定当前备份集生成路径，若未指定，则在默认备份路径中生成备份集路径。

2. 应用举例

【例 5-5】对数据库增加表空间 TBS，对其进行完全备份，备份名为 TBS_BAK1。

（1）创建 TBS 表空间。

在 DM 服务查看器中启动 DMServiceDMSERVER 服务，停止 DMServiceDMSERVERRES 服务。

```
SQL>CREATE TABLESPACE TBS DATAFILE 'C:\dmdbms\data\DAMENG\TBS1.DBF' SIZE 64;
```

（2）备份 TBS 表空间。

```
SQL>BACKUP TABLESPACE TBS FULL TO TBS_BAK1 BACKUPSET
'c:\dmdbms\data\DAMENG\bak\ TBS_BAK1' BACKUPINFO 'TBS 的联机完全备份';
```

备份完成后，在 c:\dmdbms\data\DAMENG\bak\TBS_BAK1 的目录下生成 TBS_BAK1.bak、TBS_BAK1.meta 等备份片文件和备份元数据文件。

【例 5-6】对表空间 TBS 进行增量备份，备份名为 TBS_BAK1_INCR。

```
SQL>BACKUP TABLESPACE TBS INCREMENT WITH BACKUPDIR 'c:\dmdbms\data\DAMENG\
bak\TBS_BAK1' BASE ON BACKUPSET 'c:\dmdbms\data\DAMENG\bak\TBS_BAK1' TO TBS_BAK1_
INCR BACKUPSET 'c:\dmdbms\data\DAMENG\bak\TBS_BAK2' BACKUPINFO 'TBS 的联机增量备份';
```

备份完成后，在 c:\dmdbms\data\DAMENG\bak\TBS_BAK2 的目录下生成 TBS_BAK2.bak、TBS_BAK2.meta 等备份片文件和备份元数据文件。

5.3.2 使用 SQL 语句还原

1. 语法格式

```
RESTORE TABLESPACE <表空间名> DATAFILE <文件编号> {,<文件编号>} | '<文件路径>' {,'<文件路径>'}  FROM BACKUPSET '<备份集路径>' [DEVICE TYPE <介质类型> [PARMS '<介质参数>']] [IDENTIFIED BY <密码>] [ENCRYPT WITH <加密算法>] [WITH BACKUPDIR '<备份目录>' {,'<备份目录>'}] [WITH ARCHIVEDIR '归档目录'{,'归档目录'}] [MAPPED FILE '<映射文件>'][TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>][NOT PARALLEL];
```

主要参数说明如下：

（1）表空间名：需要还原的表空间名称。

（2）文件编号：表空间中指定还原数据文件的编号，对应动态视图 V$DATAFILE 中 ID 列的值。

（3）文件路径：表空间中指定还原数据文件路径名或镜像文件名，对应动态视图 V$DATAFILE 中 PATH 或者 MIRROR_PATH 列的值；也可以仅指定数据文件名称（相对路径），当与表空间中数据文件匹配时，会使用 SYSTEM 目录补齐。

（4）备份集路径：表空间备份时指定的备份集路径，备份集可以是表空间级或者库级备份。

（5）备份目录：搜集备份文件的目录。

（6）归档目录：还原时，搜集归档文件的目录。

2. 应用举例

【例 5-7】对表空间 TBS 使用完全备份 TBS_BAK1 进行还原。

（1）使表空间脱机：

```
SQL>ALTER TABLESPACE TBS OFFLINE;
```

（2）进行还原：

```
SQL>RESTORE TABLESPACE TBS FROM BACKUPSET 'c:\dmdbms\data\DAMENG\
bak\TBS_BAK1';
```

（3）使表空间联机：

```
SQL>ALTER TABLESPACE TBS ONLINE;
```

【例 5-8】对表空间 TBS 使用增量备份 TBS_BAK1_INCR 进行还原。

首先使表空间脱机，然后进行还原，最后使表空间联机。还原时既要指定增量备份的目录，还要指定备份链路中完全备份的目录。

（1）使表空间脱机：

```
SQL>ALTER TABLESPACE TBS OFFLINE;
```

（2）进行还原：

```
SQL>RESTORE TABLESPACE TBS FROM BACKUPSET 'c:\dmdbms\data\DAMENG\
bak\TBS_BAK2' WITH BACKUPDIR 'c:\dmdbms\data\DAMENG\bak\TBS_BAK1';
```

（3）使表空间联机：

```
SQL>ALTER TABLESPACE TBS ONLINE;
```

【例 5-9】从库级备份集中还原出指定表空间 TBS。

（1）目前还没有一个完全库备份级中包含 TBS 表空间，因此先创建一个完全库备份，再基于该库备份进行表空间还原操作。

```
SQL>BACKUP DATABASE FULL BACKUPSET 'c:\dmdbms\data\DAMENG\bak\ DMBAK3';
```

（2）使表空间脱机：

```
SQL>ALTER TABLESPACE TBS OFFLINE;
```

（3）进行还原：

```
SQL>RESTORE TABLESPACE TBS FROM BACKUPSET 'c:\dmdbms\data\DAMENG\ bak\DMBAK3'
```

```
WITH BACKUPDIR 'c:\dmdbms\data\DAMENG\bak\DMBAK3';
```

（4）使表空间联机：

```
SQL>ALTER TABLESPACE TBS ONLINE;
```

【例 5-10】从库级备份集 DMBAK3 中还原出指定数据文件 TBS1.DBF。

（1）使表空间脱机：

```
SQL>ALTER TABLESPACE TBS OFFLINE;
```

（2）进行还原：

```
SQL>RESTORE TABLESPACE TBS DATAFILE 'TBS1.DBF' FROM BACKUPSET
'c:\dmdbms\data\DAMENG\bak\DMBAK3' WITH BACKUPDIR 'c:\dmdbms\data\ DAMENG\bak\DMBAK3';
```

（3）使表空间联机：

```
SQL>ALTER TABLESPACE TBS ONLINE;
```

3. 使用说明

（1）支持从库级备份集中还原出指定表空间，以及从库级和表空间级备份集中还原出表空间中指定数据文件。

（2）备份集路径是指备份集所在目录，其中应包含完整备份数据，包括元数据文件（.meta）和备份片文件（.bak）。

（3）还原前应该先将还原的表空间脱机，只能还原用户表空间。

（4）该功能在配置本地归档后才起作用。

（5）数据守护环境下不允许进行表空间还原。

（6）MOUNT 状态下不允许进行表空间还原。

（7）MPP 和 RAC 环境均不允许进行表空间还原。

（8）若指定 NOT PARALLEL，则认为并行备份不使用并行还原，默认并行备份执行并行还原，忽略非并行备份集还原。

5.4 表备份还原

5.4.1 使用 SQL 语句备份

1. 语法格式

```
BACKUP TABLE <表名> [WITH INDEX [索引名称{,索引名称}]] [TO <备份名>] BACKUPSET '<备份集路径>' [DEVICE TYPE <介质类型> [PARMS '<介质参数>']] [BACKUPINFO '<备份集描述>'] [MAXPIECESIZE <备份片限制大小>] [IDENTIFIED BY <密钥>[WITH ENCRYPTION<TYPE>][ENCRYPT WITH <加密算法>]] [COMPRESSED [LEVEL <压缩级别>]] [TRACE FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>];
```

主参数说明如下：

（1）表名：需要备份的用户表的名称，只能备份用户表。

（2）WITH INDEX：如果指定 WITH INDEX，那么还原操作的时候会在还原数据之前创建二级索引，否则在之后创建。

（3）备份名：指定生成备份名称。若未指定，则系统随机生成，默认备份名格式为 DB_BTREE_数据库名_备份时间。

（4）备份集路径：指定当前备份集生成路径，若未指定，则在默认备份路径中生成备份集路径。

2. 应用举例

【例 5-11】创建 TTBAK 表，并进行备份，备份名为 TTBAKSET。

（1）创建 TTBAK 表，并插入一条数据：

```
SQL>CREATE TABLE OTHER.TTBAK(A INT,B INT);
SQL>INSERT INTO OTHER.TTBAK VALUES(100,100);
SQL>COMMIT;
```

（2）备份 TTBAK 表：

```
SQL>BACKUP TABLE OTHER.TTBAK TO TTBAKSET BACKUPSET 'c:\dmdbms\
data\DAMENG\bak\TTBAKSET' BACKUPINFO 'TTBAK 表的备份';
```

3. 使用说明

（1）仅支持对普通用户表进行备份，包括 LIST 表。

（2）当备份数据超过限制大小时，会生成新的备份文件，新的备份文件名是初始文件名后加文件编号。

（3）表备份时，其所属表空间必须处于联机状态。

（4）数据守护环境下，仅允许在主机上进行表备份。

（5）MOUNT 状态下不允许进行表备份。

（6）MPP 和 RAC 环境下均不允许进行表备份。

5.4.2　使用 SQL 语句还原

1. 语法格式

```
RESTORE TABLE [<表名>][STRUCT] [WITH INDEX | WITH INDEX DATA | WITHOUT INDEX]
[WITH CONSTRAINT|WITHOUT CONSTRAINT] FROM BACKUPSET'<备份集路径>' [DEVICE TYPE <
介质类型> [PARMS '<介质参数>']] [IDENTIFIED BY <密码>] [ENCRYPT WITH <加密算法>] [TRACE
FILE '<trace 文件名>'] [TRACE LEVEL <trace 日志级别>];
```

主要参数说明如下：

（1）表名：需要还原的表名称。

（2）备份集路径：表空间备份时指定的备份集路径。

2. 应用举例

【例 5-12】创建 TTREP 表，并使用备份集 TTBAKSET 进行还原。

（1）创建 TTREP 表，并插入一条数据。

```
SQL>CREATE TABLE OTHER.TTREP(A INT, B INT);
SQL>INSERT INTO OTHER.TTREP VALUES(1, 1);
SQL>COMMIT;
SQL>SELECT * FROM OTHER.TTREP;
行号          A             B
---------- ----------- --------------------
1             1             1
```

（2）对 TTREP 表使用 TTBAKSET 备份集还原数据。

先查看 TTBAK 表中的数据内容：

```
SQL>SELECT * FROM OTHER.TTBAK;
行号          A             B
---------- ----------- --------------------
1             100           100
```

再用 TTBAK 表的备份集来还原 TTREP 表：

```
SQL>RESTORE TABLE OTHER.TTREP FROM BACKUPSET 'c:\dmdbms\data\
DAMENG\bak\TTBAKSET';
SQL>SELECT * FROM OTHER.TTREP;
行号          A             B
---------- ----------- --------------------
1             100           100
```

这个例子说明，还原后，表的数据被备份集中的数据覆盖。

【例 5-13】删除 TTBAK 表，并从 TTBAKSET 备份集中还原 TTBAK 表。

（1）删除 TTBAK 表：

```
SQL>DROP TABLE OTHER.TTBAK;
```

（2）恢复 TTBAK 表结构：

```
SQL>RESTORE TABLE STRUCT FROM BACKUPSET 'c:\dmdbms\data\DAMENG\ bak\TTBAKSET';
```

（3）还原 TTABK 表数据：

```
SQL>RESTORE TABLE WITH INDEX DATA FROM BACKUPSET
'c:\dmdbms\data\DAMENG\bak\TTBAKSET';
SQL>SELECT * FROM OTHER.TTBAK;
行号          A             B
---------- ----------- --------------------
1             100           100
```

这个例子说明，在没有数据表的情况下，要先还原表结构，再还原表数据。

3. 使用说明

（1）备份集路径指备份集所在目录，其中应包含完整备份数据，包括元数据文件（.meta）和备份片文件（.bak）。

（2）还原目标表名可指定，也可不指定；若指定，则必须存在且表定义必须与备份表严格一致；若不指定，则使用备份集中记录的备份表作为还原目标表。

（3）若指定 WITHOUT INDEX，则还原后目标表中无索引信息，即使目标表本来存在，也会被删除；若未指定，则在数据还原后重新创建备份集中索引，目标表原来的索引信息会被删除。

（4）若指定 WITHOUT CONSTRAINT，则还原后目标表中无约束信息，聚集主键除外，即使目标表本来存在，也会被删除；若未指定，则在数据还原后重新创建备份集中约束，目标表原来的约束信息会被删除。

（5）还原前目标表所在的表空间必须处于联机状态。

（6）数据守护环境下不允许进行表还原。

（7）MOUNT 和 SUSPEND 状态下不允许进行表还原。

（8）MPP 和 RAC 环境下均不允许进行表还原。

（9）若在语句中指定 STRUCT 关键字，则执行表结构还原。表结构还原会根据备份集中备份表的还原要求，对目标表定义进行校验，并删除目标表中已存在的二级索引和约束；若未指定 WITHOUT CONSTRAINT，则会将备份集中的 CHECK 约束在目标表上重建。

（10）若不指定 STRUCT 关键字，则执行表数据还原。表数据还原仅会在目标表定义与备份表一致且不存在二级索引和约束的情况下执行；若还原时未指定 WITHOUT CONSTRAINT，则允许目标表中存在与备份表中相同的 CHECK 约束。

（11）若在未指定 STRUCT 的情况下，执行还原出现存在二级索引或冗余约束的错误；或者不指定目标表时，存在目标不存在的情况，可先执行 STRUCT 还原后，再继续执行实际数据的还原。

（12）数据还原（非 STRUCT）时指定 WITH INDEX DATA，会将备份集中备份的二级索引数据还原到目标索引中，不考虑该二级索引是用户创建的还是由于创建约束而系统自动创建的；结构还原（STRUCT）时，若指定 WITH INDEX DATA，则会在目标表中将待还原二级索引数据对应的二级索引重建。

（13）WITH/WITHOUT INDEX 用于指定还原数据后是否重建二级索引，默认重建；WITH INDEX DATA 是指使用备份集中备份的二级索引还原相应的二级索引，不指定时直接在聚集索引数据基础上重建；WITH/WITHOUT CONSTRAINT 指定值还原数据后是否在目标表上重建约束，默认重建。对于备份表中存在的无效索引和约束，仅备份创建语句，还原时并不创建，用户如有需要，可通过相关动态视图或 DMRMAN 工具查看语句后手动创建。但是若指定 WITH INDEX DATA，且备份时备份了无效索引，那么还原时会将备份的无效索引在目标表中还原，且状态仍处于无效。

5.5 逻辑备份与还原

5.5.1 逻辑备份

dexp 工具可以对本地或者远程数据库进行数据库级、用户级、模式级和表级的逻辑备份。备份的内容非常灵活，可以选择是否备份索引、数据行和权限，是否忽略各种约束（外键约束、非空约束、唯一约束等），在备份前可以选择生成日志文件，记录备份的过程以供查看。

1. 语法格式

逻辑备份的语法格式如下：

```
DEXP KEYWORD=VALUE …
```

各关键字的含义与用法见表 5-1，当关键字为 USERID 时，“USERID=”可省略。

表 5-1 关键字的含义与用法

关 键 字	含 义	备 注
USERID	用户名/口令@主机名:端口号#证书路径，如 SYSDBA/SYSDBA@server:5236#ssl_path@ssl_pwd	必选参数，其中主机名、端口号和证书路径为可选项，如果不指定，则使用默认值
FILE	用于指定导出文件名称。如果不选用该参数，则导出文件名为 DEXP.DMP 可以包含多个文件，用逗号分隔。文件名可以包含通配符%U，用于作为自动扩充文件的文件名模板。%U 表示 2 个字符宽度的数字，由系统自动生成，起始为 01	可选参数
FILESIZE	此参数用于指定单个导出文件大小的上限。可以按字节[B]、K[B]、M[B]、G[B]的方式指定大小，超出时自动增加新文件	可选参数
DIRECTORY	直接路径（N） 此参数用于指定导出文件及日志文件生成的目录，默认为 dexp 所在的目录。如果 FILE 和 LOG 参数指定的文件包含生成路径，则 FILE 和 LOG 参数中指定的路径将替代 DIRECTORY 所指定的路径，如果 FILE 和 LOG 参数指定的文件未包含路径信息，则文件将被生成到 DIRECTORY 指定的目录下。	可选参数
FULL	导出整个数据库（N）	可选参数
OWNER	所有者用户名列表，用户希望导出哪个用户的对象	可选参数
SCHEMAS	导出的模式列表	可选参数
TABLES	表名列表，指定导出的 table 名称	可选参数

（续表）

关 键 字	含　义	备　注
QUERY	用于对导出表的数据进行过滤的条件 例如，Tables=info query=”where condition”，通过 query 所制定的过滤条件，对表 info 进行导出	可选参数
PARALLEL	用于指定导出的过程中所使用的线程数目	可选参数
EXCLUDE	忽略指定的对象，包括 CONSTRAINTS、INDEXES、ROWS、TRIGGERS、GRANTS、TABLES。忽略表时，使用 TABLES:INFO 格式，如果使用 TABLES 方式导出，则指定的 TABLES 将被忽略	如 EXCLUDE=(CONSTRAINTS,INDEXES,TRIGGERS,GRANTS)EXCLUDE=TABLES：table1,table2
CONSTRAINTS	导出约束（Y）	可选参数
GRANTS	导出权限（N）	可选参数
INDEXES	导出索引（Y）	可选参数
TRIGGERS	导出触发器（Y）	可选参数
ROWS	导出数据行（Y）	可选参数
LOG	屏幕输出的日志文件	可选参数
NOLOGFILE	不使用日志文件（N）	可选参数
PARFILE	参数文件名，如果 exp 的参数很多，可以存为参数文件	可选参数
FEEDBACK	每 x 行显示进度（0）	可选参数
COMPRESS	是否压缩导出数据文件（N|Y）	可选参数，默认为 N
ENCRYPT	导出数据是否加密（N）	可选参数
ENCRYPT_PASSWORD	导出数据的加密密钥	可选参数，和 ENCRPY 同时使用
ENCRYPT_NAME	导出数据的加密算法	可选参数，和 ENCRPY、ENCRPY_ PASSWORD 同时使用
DROP	导出表后是否将表删除（N）	可选参数
DESCRIBE	导出数据文件的描述信息，记录在数据文件中	可选参数
NOLOG	屏幕上不显示日志信息（N）	可选参数
LOCAL	MPP 环境下，是否使用 LOCAL 方式登录（N）	可选参数
HELP	显示帮助信息	可选参数

2. 应用举例

【例 5-14】备份整个数据库。

（1）创建备份目录。

创建 c:\dmdbms\data\ DAMENG\bak\LOGIC 目录用于存储备份结果。

（2）备份整个数据库。

```
C:\dmdbms\bin>DEXP USERID=SYSDBA/SYSDBA
DIRECTORY=c:\dmdbms\data\DAMENG\bak\LOGIC FILE=db_all.dmp LOG=db_all.log FULL=Y
```

在备份过程中最好使用 DIRECTORY 参数指定路径，如果不指定路径，并且 FILE 和 LOG 参数都没有指定路径，则程序将根据当前的运行目录来设置相应的备份路径。

【例 5-15】备份单个用户的全部数据。将 DMHR 用户的数据全部备份。

```
C:\dmdbms\bin>DEXP SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\DAMENG\bak\ LOGIC FILE=own_dmhr.dmp LOG=own_dmhr.log OWNER=dmhr
```

通过 OWNER 参数指定被备份的用户。

【例 5-16】备份用户的一个模式的全部数据。将 SYSDBA 用户的 OTHER 模式下的数据全部备份。

```
C:\dmdbms\bin>DEXP SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\DAMENG\ bak\LOGIC FILE=sch_other.dmp  LOG=sch_other.log SCHEMAS=OTHER
```

用 SCHEMAS 参数指定被备份的模式。一般情况下用户与模式备份是相同的，但是用户可以包含多个模式，在这种情况下模式的备份是用户备份的一个子集。

【例 5-17】备份多个表的全部数据。将 OTHER 模式下 TTBAK 和 TTREP 表的全部数据备份。

```
C:\dmdbms\bin>DEXP SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\DAMENG\bak\ LOGIC FILE=tab_ttbak_ttrep.dmp LOG=tab_ttbak_ttrep.log TABLES=(other.ttbak,other.ttrep)
```

备份表数据时，必须指定模式名和表名，备份多个表数据时，表之间用逗号隔开，表名之间不加空格，多个表可以加括号，也可以不加。

【例 5-18】备份时进行压缩和加密。备份整个数据库，并压缩和加密，密码为 dmabc。

```
C:\dmdbms\bin>DEXP USERID=SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\ DAMENG\ bak\ LOGIC FILE=db_all_e.dmp LOG=db_all_ e.log FULL=Y ENCRYPT=Y ENCRYPT_PASSWORD='dmabc'
```

如果加密备份文件，则默认先压缩再加密。

【例 5-19】备份时使用参数文件。将 OTHER 模式下的 TTBAK 表数据备份，每备份 1 条记录显示一次进度。

（1）编辑参数文件 para.txt：

```
FILE=tab_ttbak_par.dmp
LOG=tab_ttbak_par.log
TABLES=other.ttbak
FEEDBACK=1
```

（2）执行备份命令：

```
C:\dmdbms\bin>DEXP USERID=SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\ DAMENG\ bak\ LOGIC PARFILE=c:\dmdbms\data\DAMENG\bak\LOGIC\para.txt
```

5.5.2 逻辑还原

DM 提供了对数据库进行逻辑还原的命令行工具 dimp.exe，位于安装目录的 bin\目录下。系统管理员可以利用它，在命令行方式下对达梦数据库进行联机逻辑还原，并支持对远程数据库的访问。

dimp 工具利用 dexp 工具生成的备份文件对数据库进行联机逻辑还原。还原的方式可以灵活选择，如是否忽略对象存在而导致的创建错误、是否导入约束、是否导入索引、导入时是否需要编译、是否生成日志等。

1. 语法格式

逻辑还原的命令格式如下：

```
DIMP KEYWORD=VALUE …
```

当关键字为 USERID 时，“USERID=”可省略，各关键字的详细说明见表 5-2。

表 5-2　dimp 的关键字

关 键 字	含　义	备　注
USERID	用户名/口令@主机名:端口号#证书路径，如：SYSDBA/SYSDBA @server:5236# ssl_path@ssl_pwd	必选参数，其中主机名、端口号和证书路径为可选项，如果不指定，使用默认值
FILE	输入文件，之前导出的文件	必选参数
DIRECTORY	导入文件所在目录	可选参数
FULL	导入整个数据库（N）	可选参数
OWNER	导入指定的用户名下的模式	可选参数
SCHEMAS	导入的模式列表	可选参数
TABLES	表名列表，指定导入的 tables 名称	可选参数
PARALLEL	用于指定导入的过程中所使用的线程数目	可选参数
IGNORE	忽略创建错误（N）。如果表已经存在，则向表中插入数据，否则报错	可选参数
EXCLUDE	忽略指定的对象（CONSTRAINTS、INDEXES、ROWS、TRIGGERS、GRANTS）	如 EXCLUDE= (CONSTRAINT)
CONSTRAINTS	导入约束　（Y）	可选参数
GRANTS	导入权限　（N）	可选参数
INDEXES	导入索引　（Y）	可选参数
TRIGGERS	导入触发器（Y）	可选参数
ROWS	导入数据行　（Y）	可选参数
LOG	屏幕输出的日志文件	可选参数
NOLOGFILE	不使用日志文件（N）	可选参数
NOLOG	屏幕上不显示日志信息（N）	可选参数
PARFILE	参数文件名，如果 imp 的参数很多，则可以保存为参数文件	可选参数
FEEDBACK	显示每 x 行（0）的进度	可选参数
COMPILE	编译过程、程序包和函数　（Y）	可选参数
INDEXFILE	将表的索引/约束信息写入指定的文件	可选参数

（续表）

关 键 字	含　义	备　注
INDEXFIRST	导入时先建索引（N）	可选参数
REMAP_SCHEMA	SOURCE_SCHEMA ： TARGET_ SCHEMA 将 SOURCE_SCHEMA 中的数据导入到 TARGET_ SCHEMA 中 （N）	可选参数
ENCRYPT_PASSWORD	数据的加密密钥	可选参数，和 dexp 中的 ENCRPY_ PASSWORD 设置的密钥一样
ENCRYPT_NAME	数据的加密算法	可选参数，和 dexp 中的 ENCRYPT_ NAME 设置的加密算法一样
SHOW/DESCRIBE	只列出文件内容 （N）	可选参数
LOCAL	MPP 环境下，是否使用 LOCAL 方式登录（N）	可选参数
HELP	显示帮助信息	可选参数

2. 应用举例

【例 5-20】还原表数据。利用 TAB_TTBAK_TTREP.DMP 文件还原 OTHER 模式下的 TTBAK 表和 TTREP 表。

（1）为了检验还原效果，先删除 TTBAK 和 TTREP 表：

```
SQL>DROP TABLE other.ttbak;
SQL>DROP TABLE other.ttrep;
```

（2）还原 TTBAK 和 TTREP 表：

```
C:\dmdbms\bin>DIMP SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\DAMENG\bak\ LOGIC FILE=tab_ttbak_ttrep.dmp LOG=tab_ttbak_ttrep_imp.log TABLES=(other.ttbak, other.ttrep) IGNORE=Y
```

（3）查询 TTBAK 表数据，检验是否还原成功：

```
SQL>SELECT * FROM other.ttbak;
行号          A              B
---------- ---------- --------------------
1             100            100
```

查询结果表明成功还原表数据。

【例 5-21】还原模式中的数据。利用 SCH_OTHER.DMP 文件还原 OTHER 模式下的数据。

（1）为了检验还原效果，先删除 OTHER 模式：

```
SQL> DROP SCHEMA OTHER CASCADE;
```

（2）还原 OTHER 模式下的数据：

```
C:\dmdbms\bin>DIMP USERID=SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\ DAMENG\bak\LOGIC FILE=sch_other.dmp LOG=sch_other_imp.log SCHEMAS=other
```

（3）查询 TTBAK 表的数据，检验是否还原成功：

```
SQL>SELECT * FROM other.ttbak;
行号          A               B
---------- ---------- --------------------
1             100             100
```

查询结果表明成功还原 OTHER 模式数据。

【例 5-22】还原用户的数据。利用 OWN_DMHR.DMP 文件还原 DMHR 用户的数据。

（1）为了检验还原效果，先删除 DMHR 用户，自然就删除了该用户的所有数据：

```
SQL>DROP USER DMHR CASCADE;
```

（2）新建 DMHR 用户：

```
SQL>CREATE USER dmhr IDENTIFIED BY DMHR12345 DEFAULT TABLESPACE dmhr;
```

（3）还原用户数据：

```
C:\dmdbms\bin>DIMP USERID=SYSDBA/SYSDBA DIRECTORY=c:\dmdbms\data\
DAMENG\bak\LOGIC FILE=own_dmhr.dmp LOG=own_dmhr_imp.log OWNER=dmhr
```

（4）查询 CITY 表的数据，检验是否还原成功：

```
SQL>SELECT * FROM dmhr.city WHERE region_id=2;
行号          CITY_ID         CITY_NAME         REGION_ID
---------- ------- --------- ----------------------------------------------------------
1             SH              上海              2
2             NJ              南京              2
```

查询结果表明成功还原 DMHR 用户数据。

第 6 章

作业管理

在管理员的工作中，有许多日常工作都是固定不变的，如定期备份数据库、定期生成数据统计报表等。这些工作既单调又费时，如果这些重复任务能够自动化完成，就可以节省大量的时间。

6.1 作业概述

1. 作业

作业是由 DM 代理程序按顺序执行的一系列指定的操作。作业中可执行的操作包括运行 DM PL/SQL 脚本、定期备份数据库、对数据库数据进行检查等。可以通过作业来执行经常重复和可调度的任务。作业按照调度的安排在服务器上执行，作业也可以由警报触发执行，并且作业可产生警报以通知用户作业的状态（成功或者失败）。

2. 步骤

每个作业由一个或多个作业步骤组成，作业步骤是作业对一个数据库或者一个服务器执行的动作。每个作业必须至少有一个作业步骤。

3. 调度

作业调度是用户定义的一个时间安排，在给定的时刻到来时，系统会启动相关的作业，依次执行作业中定义的步骤。调度可以是一次性的，也可以是周期性的。

4. 警报

警报是系统中发生的某种事件，如发生了特定的数据库操作，或出错信号，或者作业的启动、执行完毕等事件。警报主要用于通知指定的操作员，以便其迅速了解系统中发生的状况。可以为警报定义产生的条件，还可以定义当警报产生时系统采取的动作，如通知操作员执行某个特定的作业等。

5. 作业准备

在创建作业之前，数据库中必须有存储作业数据的系统表，这些系统表有 SYSJOBS、SYSJOBSTEPS、SYSJOBSCHEDULES、SYSMAILINFO、SYSJOBHISTORIES、SYSALERTHISTORIES、SYSOPERATORS、SYSALERTS、SYSALERTNOTIFICATIONS。在《达梦数据库作业操作手册》中有这些表的定义和创建方法。一个简单的办法是在管理工具中初始化代理完成作业准备工作。在管理工具中，右击“代理”节点，在弹出的快捷菜单中选择“创建代理环境”选项，在“代理”节点下创建“作业”、“警报”、“操作员”三个子节点，作业准备工作就完成了。

6.2　通过系统过程管理作业

6.2.1　创建作业

1. 语法格式

```
SP_CREATE_JOB (
   JOB_NAME             VARCHAR(128),
   ENABLED          INT,
   ENABLE_EMAIL         INT,
   EMAIL_OPTR_NAME      VARCHAR(128),
   EMAIL_TYPE           INT,
   ENABLED_NETSEND      INT,
   NETSEND_OPTR_NAME    VARCHAR(128),
   NETSEND_TYPE         INT,
   DESCRIBE             VARCHAR(8187)
)
```

参数说明如下：

（1）JOB_NAME：作业名称，必须是有效的标识符，同时不能是 DM 关键字。作业不能重名，重名则报错。

（2）ENABLED：作业是否启用。1 表示启用；0 表示不启用。

（3）ENABLE_EMAIL：作业是否开启邮件系统。1 表示是；0 表示否。如果开启，那么该作业相关的一些日志会通过邮件通知操作员；不开启就不会发送邮件。

（4）EMAIL_OPTR_NAME：指定操作员名称。如果开启了邮件通知功能，邮件会发送给该操作员。在创建时系统会检测这个操作员是否存在，如果不存在则报错。

（5）EMAIL_TYPE：在开启了邮件发送之后，在什么情况下发送邮件。情况分为三种：0、1、2。0 表示在作业执行成功后发送；1 表示在作业执行失败后发送；2 表示在作业执行结束后发送。

（6）ENABLED_NETSEND：作业是否开启网络发送。1 表示是；0 表示否。如果开启，那么这个作业相关的一些日志会通过网络发送通知操作员；如果不开启，则不会通知。

（7）NETSEND_ OPTR_NAME：指定操作员名称。如果开启了网络信息通知功能，则会通过网络发送来通知该操作员。在创建时系统会检测这个操作员是否存在，如果不存在则报错。

（8）NETSEND_TYPE：在开启了网络发送之后，在什么情况下发送网络信息。这个情况也有三种，和上面的 EMAIL_TYPE 是完全一样的。网络发送功能只有 Windows 早期版本上才支持（如 Windows 2000/XP），且一定要开启 MESSAGER 服务。Windows 7、8 系统因为取消了 MESSAGER 服务，所以也不支持该功能。

（9）DESCRIBE：作业描述信息，最长 500 个字节。

2. 应用举例

【例 6-1】创建 INSERTTTBAK 作业。不开启邮件，不指定操作员，不开启网络发送。

```
SQL>SP_CREATE_JOB('INSERTTTBAK',1,0,'',0,0,'',0,'向 TTBAK 表插入数据');
```

6.2.2 启动作业配置

1. 语法格式

```
SP_JOB_CONFIG_START (
   JOB_NAME                VARCHAR(128)
)
```

JOB_NAME 指要配置的作业的名称。执行时会检测这个作业是否存在，如果不存在则报错。

2. 应用举例

【例 6-2】开始配置 INSERTTTBAK。

```
SQL>SP_JOB_CONFIG_START('INSERTTTBAK');
```

6.2.3　配置作业步骤

1. 语法格式

```
SP_ADD_JOB_STEP (
   JOB_NAME          VARCHAR(128),
   STEP_NAME         VARCHAR(128),
   TYPE              INT,
   COMMAND           VARCHAR(8187),
   SUCC_ACTION       INT,
   FAIL_ACTION       INT,
   RETRY_ATTEMPTS    INT,
   RETRY_INTERVAL    INT,
   OUTPUT_FILE_PATH  VARCHAR(256),
   APPEND_FLAG       INT
)
```

参数说明如下：

（1）JOB_NAME：作业的名称，表示正在给哪一个作业增加步骤，这个参数必须为上面调用 SP_JOB_CONFIG_START 函数时指定的作业名，否则系统会报错，同时系统会检测这个作业是否存在，不存在也会报错。

（2）STEP_NAME：表示增加的步骤名，必须是有效的标识符，同时不能是 DM 关键字。同一个作业不能有两个同名的步骤，创建时会检测这个步骤是否已经存在，如果存在则报错。

（3）TYPE：步骤的类型。取值为 0、1、2、3、4、5 和 6。

0 表示执行一段 SQL 语句或者语句块。

1 表示执行备份还原 I。

2 表示重组数据库。

3 表示更新数据库的统计信息。

4 表示执行 DTS（数据迁移）。

5 表示执行备份还原 II。

6 表示执行基于备份集的备份还原。

（4）COMMAND：指定不同步骤类型（TYPE）下，步骤在运行时所执行的语句。它不能为空。

当 TYPE=0 时，指定要执行的 SQL 语句或者语句块。如果要指定多条语句，在语句之间必须用分号隔开。不支持多条 DDL 语句一起执行，否则在执行时可能会报出不可预知的错误信息。

当 TYPE=1 时，指定的是一个字符串。该字符串由三部分组成：[备份模式][备份压缩类

型][base_dir,…,base_dir|bakfile_path]。三部分详细介绍如下。

第一部分是一个字符，表示备份模式。0 表示完全备份；1 表示增量备份。如果第一个字符不是这两个值中的一个，系统会报错。

第二部分是一个字符，表示备份时是否进行压缩。0 表示不压缩；1 表示压缩。

第三部分是一个文件路径，表示备份文件的路径。路径命令有具体的格式，分为以下两种。

对于增量备份，因为它必须要指定一个或者多个基备份路径，每个路径之间需要用逗号隔开，之后是备份路径，基备份路径与备份路径需要用“|”隔开，如果不指定备份路径，则不需要指定“|”，同时系统会自动生成一个备份路径。例如，01E:\base_bakdir1，base_bakdir2|bakdir。

对于完全备份，因为不需要指定基备份，所以不需要“|”符号，可以直接在第三个字节开始指定备份路径。例如，01E:\bakdir。如果不指定备份路径，则系统会自动生成一个备份路径。

当 TYPE 是 2、3 或 4 时，要执行的语句就是由系统内部根据不同类型生成的不同语句或者过程。

当 TYPE=5 时，指定的是一个字符串。该字符串由六部分组成：[备份模式][备份压缩类型][备份日志类型][备份并行类型][预留][base_dir,…,base_dir | bakfile_path | parallel_file]。六部分详细介绍如下。

第一部分是一个字符，表示备份模式。0 表示完全备份；1 表示增量备份。如果第一个字符不是这两个值中的一个，系统会报错。

第二部分是一个字符，表示备份时是否进行压缩。0 表示不压缩；1 表示压缩。

第三部分是一个字符，表示是否备份日志。0 表示备份；1 表示不备份。

第四部分是一个字符，表示是否并行备份。0 表示普通备份；1 表示并行备份，并行备份映射放到最后，以“|”分割。

第五部分是一个保留字符，用 0 填充。

第六部分是一个文件路径，表示备份文件的路径。

当 TYPE=6 时，指定的是一个字符串。该字符串由九部分组成：[备份模式][备份压缩类型][备份日志类型][备份并行数][USE PWR][MAXPIECESIZE][RESV1][RESV2][base_dir,…,base_dir | bakfile_dir]。九部分详细介绍如下。

第一部分是一个字符，表示备份模式。0 表示完全备份；1 表示增量备份；3 表示归档备份。如果第一个字符不是这三个值中的一个，系统会报错。

第二部分是一个字符，表示备份时是否进行压缩。0 表示不压缩；1 表示压缩。

第三部分是一个字符，表示是否备份日志。0 表示备份；1 表示不备份。

第四部分是一个字符，表示并行备份并行数。取值为 0～9。其中，0 表示不进行并行备份；1 表示使用并行数默认值 4；2～9 表示并行数。

第五部分为一个字符，表示并行备份时，是否使用 USE PWR 优化增量备份。0 表示不使用；1 表示使用。

第六部分为一个字符，表示备份片大小的上限（MAXPIECESIZE）。0 表示默认值，1～

9 表示依次表示备份片的大小，1 表示 128MB 备份片大小，9 表示 32GB 的备份片大小。

第七部分为一个字符，表示是否在备份完归档后，删除备份的归档文件。0 表示不删除；1 表示删除。

第八部分是一个保留字符，用 0 填充。

第九部分是一个文件路径，表示备份文件的路径。

（5）SUCC_ACTION 指定步骤执行成功后，下一步该做什么事。取值为 0、1 或 3。

0 表示执行下一步。

1 表示报告执行成功。

3 表示返回第一个步骤继续执行。

（6）FAIL_ACTION 指定步骤执行失败后，下一步该做什么事。取值为 0、2 或 3。

0 表示执行下一步。

2 表示报告执行失败。

3 表示返回第一个步骤继续执行。

（7）RETRY_ATTEMPTS：表示当步骤执行失败后，需要重试的次数，取值为 0～100。

（8）RETRY_INTERVAL：表示在每两次步骤执行重试之间的间隔时间，不能大于 10 秒钟。

（9）OUTPUT_FILE_PATH：表示步骤执行时输出文件的路径。该参数已废弃，没有实际意义。

（10）APPEND_FLAG：输出文件的追写方式。如果指定输出文件，那么这个参数表示在写入文件时是否从文件末尾开始追写。1 表示是；0 表示否。如果是 0，那么从文件指针当前指向的位置开始追写。

2. 应用举例

【例 6-3】下面的语句为作业 INSERTTTBAK 增加了步骤 STEP1，STEP1 的任务是向 OTHER 模式下的 TTBAK 表中插入一条记录。

```
SQL>SP_ADD_JOB_STEP('INSERTTTBAK', 'STEP1', 0, 'INSERT INTO OTHER.TTBAK VALUES (200,200);', 0, 0, 0, 0, NULL, 0);
```

STEP1 指定的是执行 SQL 语句，成功或失败后执行下一个步骤，失败后不重新执行，两个步骤之间间隔为 0。

6.2.4　配置作业调度

1. 语法格式

```
SP_ADD_JOB_SCHEDULE (
  JOB_NAME              VARCHAR(128),
  SCHEDULE_NAME         VARCHAR(128),
```

```
    ENABLE                  INT,
    TYPE                    INT,
    FREQ_INTERVAL           INT,
    FREQ_SUB_INTERVAL       INT,
    FREQ_MINUTE_INTERVAL    INT,
    STARTTIME               VARCHAR(128),
    ENDTIME                 VARCHAR(128),
    DURING_START_DATE       VARCHAR(128),
    DURING_END_DATE         VARCHAR(128),
    DESCRIBE                VARCHAR(500)
);
```

参数说明如下：

（1）JOB_NAME：作业名称，指定要给该作业增加调度，这个参数必须是配置作业开始时指定的作业名，否则报错，同时系统还会检测这个作业是否存在，如果不存在也会报错。

（2）SCHEDULE_NAME：待创建的调度名称，必须是有效的标识符，同时不能是 DM 关键字。指定的作业不能创建两个同名的调度，创建时会检测这个调度是否已经存在，如果存在则报错。

（3）ENABLE：表示调度是否启用，布尔类型。1 表示启用；0 表示不启用。

（4）TYPE：指定调度类型。取值为 0、1、2、3、4、5、6、7、8。

0 表示指定作业只执行一次。

1 表示按天的频率来执行。

2 表示按周的频率来执行。

3 表示在一个月的某一天执行。

4 表示在一个月的第一周的第几天执行。

5 表示在一个月的第二周的第几天执行。

6 表示在一个月的第三周的第几天执行。

7 表示在一个月的第四周的第几天执行。

8 表示在一个月的最后一周的第几天执行。

当 TYPE=0 时，其执行时间由参数 DURING_START_DATE 指定。

（5）FREQ_INTERVAL：与 TYPE 有关，表示不同调度类型下的发生频率。

当 TYPE=0 时，这个值无效，系统不做检查。

当 TYPE=1 时，表示每几天执行，取值为 1～100。

当 TYPE=2 时，表示每几个星期执行，取值范围没有限制。

当 TYPE=3 时，表示每几个月中的某一天执行，取值范围没有限制。

当 TYPE=4 时，表示每几个月的第一周执行，取值范围没有限制。

当 TYPE=5 时，表示每几个月的第二周执行，取值范围没有限制。

当 TYPE=6 时，表示每几个月的第三周执行，取值范围没有限制。

当 TYPE=7 时，表示每几个月的第四周执行，取值范围没有限制。

当 TYPE=8 时，表示每几个月的最后一周执行，取值范围没有限制。

（6）FREQ_SUB_INTERVAL：与 TYPE 和 FREQ_INTERVAL 有关，表示不同 TYPE 的执行频率，在 FREQ_INTERVAL 基础上，继续指定更为精准的频率。

当 TYPE=0 或 1 时，这个值无效，系统不做检查。

当 TYPE=2 时，表示某一个星期的星期几执行，可以同时选中七天中的任意几天。取值为 1～127。具体如何取值，请用户参考如下规则。因为每周有七天，所以 DM 数据库系统内部用七位二进制来表示选中的日子。从最低位开始算起，依次表示周日、周一、……、周五、周六。选中周几，就将该位置 1，否则置 0。例如，选中周二和周六，7 位二进制就是 1000100，转化成十进制就是 68，所以 FREQ_SUB_INTERVAL 就取值 68。

当 TYPE=3 时，表示将在一个月的第几天执行，取值为 1～31。

当 TYPE 为 4、5、6、7 或 8 时，都表示将在某一周内第几天执行，取值为 1～7，分别表示从周一到周日。

（7）FREQ_MINUTE_INTERVAL：表示一天内每隔多少分钟执行一次，有效值为 1～1440，单位为分钟。

（8）STARTTIME：定义作业被调度的起始时间，必须是有效的时间字符串，不可以为空。

（9）ENDTIME：定义作业被调度的结束时间，可以为空。如果不为空，指定的必须是有效的时间字符串，同时必须要在 STARTTIME 时间之面。

（10）DURING_START_DATE：指定作业被调度的起始日期，必须是有效的日期字符串，不可以为空。

（11）DURING_END_DATE：指定作业被调度的结束日期，可以为空，DURING_END_DATE 和 ENDTIME 都为空时，调度活动会一直持续下去。如果不为空，必须是有效的日期字符串，同时必须在 DURING_START_ DATE 日期之后。

（12）DESCRIBE：表示调度的注释信息，最大长度为 500 个字节。

2. 应用举例

【例 6-4】为作业 INSERTTTBAK 增加名为 SCH1 的调度。

```
SQL>SP_ADD_JOB_SCHEDULE('INSERTTTBAK','SCH1', 1, 1, 1, 0, 1, CURTIME, '23:59:59',CURDATE, NULL, '一个测试调度');
```

在上面的例子中，为作业 INSERTTTBAK 创建了一个新的调度，调度名为 SCH1；ENABLE 为 1，即启用这个调度；其调度类型 TYPE 为 1，表示只执行一次；FREQ_INTERVAL 为 1，说明每天都要执行；在这种类型下 FREQ_SUB_INTERVAL 参数就不会检查，随机写 0；FREQ_MINUTE_ INTERVAL 指定的是 1，说明每隔一分钟就执行一次；STARTTIME 指定从当前时间开始，CURTIME 表示系统当前时间；ENDTIME 指定 23:59:59，表示每天都执行到这个时间为止；DURING_START_DATE 为调度的起始日期，CURDATE 表示系统当前日期；DURING_END_DATE 指定为 NULL，表示这个调度指定的日期为从开始执行那一刻起，永不停止；DESCRIBE 指定为'一个测试调度'，这是对这个调度的注释。

6.2.5 提交作业配置

1. 语法格式

```
SP_JOB_CONFIG_COMMIT (
   JOB_NAME                VARCHAR(128)
)
```

JOB_NAME 指待结束配置的作业的名称。

2. 应用举例

【例 6-5】提交作业，并查看作业执行的效果。

（1）在提交作业之前，为了便于查看作业执行前后的效果，清空 TTBAK 表中的数据，这一步非必要。

```
SQL>DELETE FROM OTHER.TTBAK;
```

（2）提交作业。

```
SQL>SP_JOB_CONFIG_COMMIT('INSERTTTBAK');
```

（3）几分钟后，查询 TTBAK 表中的数据，检查作业执行情况。

```
SQL>SELECT * FROM OTHER.TTBAK;
行号          A              B
---------- ---------- --------------------
1             200            200
2             200            200
3             200            200
4             200            200
5             200            200
```

查询结果表明作业执行成功。

6.2.6 其他作业管理

这里简要介绍其他作业管理操作，系统过程的详细语法请参考《达梦数据库作业操作手册》。

1. 查看作业日志

创建的每一个作业信息都存储在作业表 SYSJOBHISTORIES 中。通过查看表 SYSJOBHISTORIES，可以看到所有已经创建的作业。

【例 6-6】查看作业日志。查看 INSERTTTBAK 作业的日志。

```
SQL>SELECT STEPNAME, STATUS, CUR_TIME FROM SYSJOB.SYSJOBHISTORIES WHERE
```

```
NAME='INSERTTTBAK' ORDER BY ID;
    行号        STEPNAME              STATUS                CUR_TIME
    ---------- -------- ------------ ------------------------------------------------------------------------------------
    1                                JOB START             2016-07-30 07:41:34.732
    2           STEP1                JOB STEP START        2016-07-30 07:41:34.732
    3           STEP1                JOB STEP END          2016-07-30 07:41:34.732
    4                                JOB END               2016-07-30 07:41:34.732
    5                                JOB START             2016-07-30 07:42:34.732
    6           STEP1                JOB STEP START        2016-07-30 07:42:34.732
    7           STEP1                JOB STEP END          2016-07-30 07:42:34.732
    8                                JOB END               2016-07-30 07:42:34.732
    9                                JOB START             2016-07-30 07:43:34.857
    10          STEP1                JOB STEP START        2016-07-30 07:43:34.857
    11          STEP1                JOB STEP END          2016-07-30 07:43:34.857
    12                               JOB END               2016-07-30 07:43:34.857
```

由于篇幅限制，这里仅列出三次作业调度过程，每一次调度包括作业启动（JOB START）、步骤启动（JOB STEP START）、步骤运行结束（JOB STEP END）、作业结束（JOB END）等四个过程。

2. 清除作业日志

因为日志记录会不断增加，越来越庞大，所以用户需要及时清理过时的日志。可以通过系统过程 SP_JOB_CLEAR_HISTORIES 清除迄今为止某个作业的所有日志记录，即删除表 SYSJOBHISTORIES 中的相关记录。如果该作业还在继续工作，那么后续会在表 SYSJOBHISTORIES 中产生该作业的新日志。

【例 6-7】清除作业日志。清除 INSERTTTBAK 作业的日志。

```
SQL>SP_JOB_CLEAR_HISTORIES('INSERTTTBAK');
```

3. 删除步骤

如果用户发现一个作业中的某个步骤不需要了，可以通过系统过程 SP_DROP_JOB_STEP 删除这个步骤。删除步骤必须在配置作业开始后才能进行，否则系统会报错，这样处理主要是为了保证作业配置的完整性。

【例 6-8】删除步骤。不需要 STEP1 步骤，删除该步骤。

```
SQL>SP_JOB_CONFIG_START('INSERTTTBAK');
SQL>SP_DROP_JOB_STEP('INSERTTTBAK', 'STEP1');
```

4. 删除调度

如果不再需要某一个调度，可以将其删除。调用的函数为 SP_DROP_JOB_SCHEDULE。删除调度前，同样需要配置作业开始后才能进行。

【例 6-9】删除调度。不需要 SCH1 调度，删除该步骤。

```
SQL>SP_DROP_JOB_SCHEDULE('INSERTTTBAK', 'SCH1');
```

5. 更改作业

有时需要修改作业参数，可以通过 SP_ALTER_JOB 系统过程来实现。

【例 6-10】停止作业。停止 INSERTTTBAK 作业，即将作业的 ENABLE 参数修改为 0，其他参数不变。

```
SQL>SP_ALTER_JOB('INSERTTTBAK',0,0,'',0,0,'',0,'向 TTBAK 表插入数据');
```

执行该系统过程后，作业就不会继续执行了。

6. 删除作业

如果一个作业已经执行完成，或者由于其他原因需要删除作业，可以调用系统过程 SP_DROP_JOB 实现。

【例 6-11】删除作业。因不再需要 INSERTTTBAK 作业，故删除该作业。

```
SQL>SP_DROP_JOB('INSERTTTBAK');
```

6.3 通过管理工具管理作业

【例 6-12】创建作业 BAKALL，目的是每周三给对数据库做一次完全备份。

步骤 1：右击“作业”节点，在弹出的快捷菜单中选择“新建作业”选项，如图 6-1 所示，弹出“新建作业”对话框，如图 6-2 所示。

图 6-1 新建作业

步骤 2：在图 6-2 中，作业名设置为“BAKALL”，作业描述为“全库定期备份”。

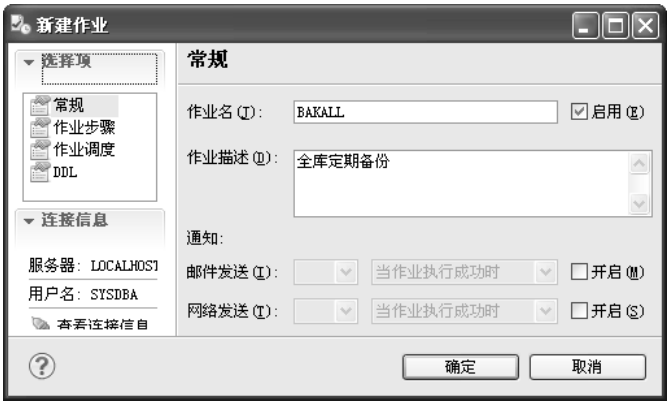

图 6-2　设置作业名称和描述

步骤 3：选择“作业步骤”选项，单击“添加”按钮，弹出修改建作业步骤对话框，如图 6-3 所示。

步骤 4：在图 6-3 中，设置步骤名称为 STEP1、步骤类型为备份数据库、备份路径为 C:\dmdbms\data\DAMENG\DBBAK，备份方式为完全备份，单击“确定”按钮返回。

图 6-3　修改作业步骤

步骤 5：选择“作业调度”选项，单击“添加”按钮，弹出“新建作业调度”对话框，如图 6-4 所示。

步骤 6：在图 6-4 中，设置调度名称为 SCH1，调度类型为反复执行，发生频率为每周三的 9:25:18 执行一次，如图 6-4 所示，单击“确定”按钮返回。

步骤 7：在图 6-2 中，继续单击“确定”按钮，完成 BAKALL 作业的创建。

步骤 8：到达指定的时间后，查看 BAKALL 作业历史信息，结果如图 6-5 所示。

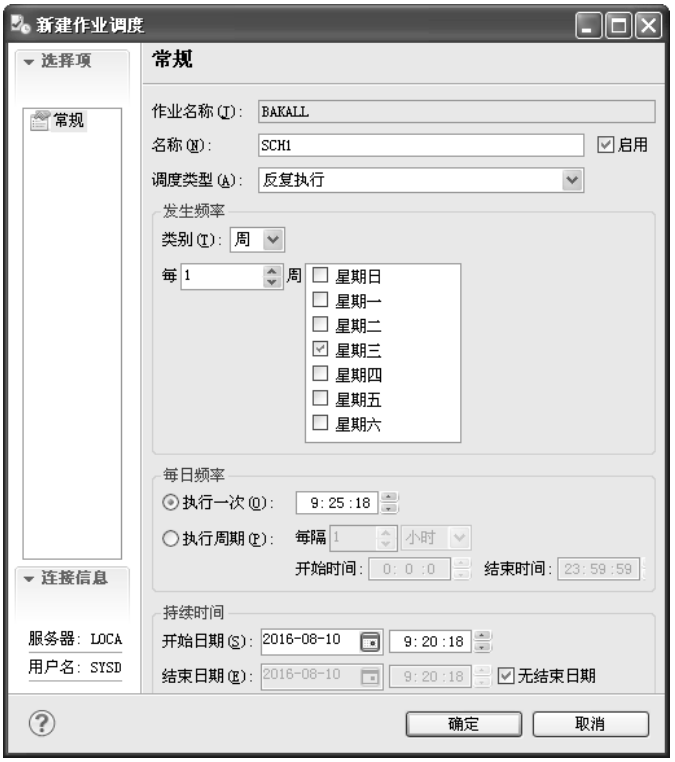

图 6-4 “新建作业调度”对话框

作业名	步骤名	状态	错误类型	错误码	..	记录时间	重试次数
BAKALL		JOB START	0	0		2016-08-10 09:25:42.988	0
BAKALL	STEP1	JOB STEP START	0	0		2016-08-10 09:25:42.988	0
BAKALL	STEP1	JOB STEP END	0	0		2016-08-10 09:25:44.879	0
BAKALL		JOB END	0	0		2016-08-10 09:25:44.879	0

图 6-5 数据表查询结果

在 C:\dmdbms\data\DAMENG\DBBAK 目录下创建了一个 DB_DAMENG_2016_08_10_09_25_42.bak 完全备份文件。

7

第 7 章

安全管理

数据库安全管理是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。安全问题不是数据库系统所独有的，计算机系统都有这个问题，只是在数据库系统中有大量数据集存放，而且为许多用户直接共享，是宝贵的信息资源，从而使安全性问题更为突出。

数据库安全保护措施是否有效是衡量数据库系统的重要指标之一。DM 的安全管理就是为保护存储在达梦数据库中的各类敏感数据的机密性、完整性和可用性提供的必要技术手段，防止对这些数据的非授权泄露、修改和破坏，并保证被授权用户能按其授权范围访问所需要的数据。这些内容属于高级系统管理范畴，是系统管理员应该掌握的内容。安全管理对信息敏感部门尤为重要，应用系统设计者和系统管理员必须熟练掌握系统的安全特性，以便建立高安全性的数据库应用系统。

7.1 权限管理

达梦数据库对用户的权限有着严格的规定，如果没有权限，用户将无法完成任何操作。用户权限有两类，即数据库权限和对象权限。数据库权限主要是指针对数据库对象的创建、删除、修改数据库对象的权限，以及对数据库备份等权限。而对象权限主要是指对数据库对象中的数据的访问权限。数据库权限一般由 SYSDBA、SYSAUDITOR 和 SYSSSO 指定，也可以由具有特权的其他用户授予。对象权限一般由数据库对象的所有者授予用户，也可由 SYSDBA 用户指定，或者由具有该对象权限的其他用户授权。

7.1.1 权限分类

1. 数据库权限

数据库权限是与数据库安全有关的最重要的权限，这类权限一般是针对数据库管理员的。数据库权限的管理主要包括权限的分配、回收和查询等操作。DM 提供了 100 余种数据库权限，表 7-1 列出了与用户有关的最重要的几种数据库权限。

表 7-1 常用数据库权限

数据库权限	说　明
CREATE TABLE	在自己的模式中创建表的权限
CREATE VIEW	在自己的模式中创建视图的权限
CREATE USER	创建用户的权限
CREATE TRIGGER	在自己的模式中创建触发器的权限
ALTER USER	修改用户的权限
ALTER DATABASE	修改数据库的权限
CREATE PROCEDURE	在自己的模式中创建存储过程的权限

对于表、视图、用户、触发器这些数据库对象，有关的数据库权限包括创建、删除和修改，相关的命令分别是 CREATE、DROP 和 ALTER。表、视图、触发器、存储程序等对象是与用户有关的，在默认情况下对这些对象的操作都是在当前用户自己的模式下进行的。如果要在其他用户的模式下操作这些类型的对象，需要具有相应的 ANY 权限。例如，要能够在其他用户的模式下创建表，当前用户必须具有 CREATE ANY TABLE 数据库权限，如果希望能够在其他用户的模式下删除表，则必须具有 DROP ANY TABLE 数据库权限。

数据库权限的授予者一般是数据库管理员。普通用户被授予了某种数据库权限及其转授权时，系统允许它把所拥有的数据库权限再授予其他用户。

2. 对象权限

对象权限主要是对数据库对象中的数据的访问权限，这类权限主要是针对普通用户的。表 7-2 列出了主要的对象权限。

表 7-2 主要的对象权限

数据库对象权限	表	视图	存储程序	包	类	类型	序列	目录	域
SELECT	✓	✓					✓		
INSERT	✓	✓							
DELETE	✓	✓							
UPDATE	✓	✓							
REFERENCES	✓								

（续表）

数据库对象权限	表	视图	存储程序	包	类	类型	序列	目录	域
DUMP	✓								
EXECUTE			✓	✓	✓	✓		✓	
READ								✓	
WRITE								✓	
USAGE									✓

SELECT、INSERT、DELETE 和 UPDATE 权限分别针对数据库对象中的数据的查询、插入、删除和修改权限。对于表和视图来说，删除操作是整行进行的，而查询、插入和修改却可以在一行的某个列上进行，所以在指定权限时，DELETE 权限只要指定所要访问的表即可，而 SELECT、INSERT 和 UPDATE 权限还可以进一步指定是对哪个列的权限。

REFERENCES 权限是指可以与一个表建立关联关系的权限，如果具有了这个权限，当前用户就可以通过自己的一个表中的外键，与对方的表建立关联。关联关系是通过主键和外键进行的，所以在授予这个权限时，可以指定表中的列，也可以不指定。

EXECUTE 权限是指可以执行存储函数、存储过程的权限。有了这个权限，一个用户就可以执行另一个用户的存储程序。

当一个用户获得另一个用户的某个对象的访问权限后，以“模式名.对象名”的形式访问这个数据库对象。一个用户所拥有的对象和可以访问的对象是不同的，这一点在数据字典视图中有所反映。默认情况下，用户可以直接访问自己模式中的数据库对象，但是要访问其他用户所拥有的对象，就必须具有相应的对象权限。

7.1.2　授予权限

1. 授予数据库权限

1）语法格式

授予数据库权限的 SQL 命令格式如下：

```
GRANT   <权限 1>{,<权限 2>}
TO <用户 1>{,<用户 2>}
[WITH ADMIN OPTION];
```

数据库权限通常指针对表、视图、用户、触发器等类型的对象具有 CREATE、ALTER、DROP 等操作能力。如果使用 ANY 修饰词，则表示对所有用户模式下的这些类型对象都具有相应操作权限。如果使用 WITH ADMIN OPTION 选项，则表示用户 1（用户 2…）获得权限后，还可以把这个权限再次授予其他用户。

2）应用举例

【例 7-1】将创建表的权限授予一个用户。以 SYSDBA 用户登录系统，将 CREATE TABLE 权限授予用户 USER1。

（1）给 USER1 授予创建表权限：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT CREATE TABLE TO USER1;
```

（2）USER1 创建 U1T1 表：

```
SQL> CONN USER1/PWORDUSER1;
SQL>CREATE TABLE u1t1
(
    id INT,
    text VARCHAR(30)
);
```

执行这些语句后，USER1 成功创建 U1T1 表。

【例 7-2】将创建表的权限授予一个用户并使用 WITH ADMIN OPTION 选项。以 SYSDBA 用户登录系统，将 CREATE TABLE 权限授予用户 USER1，USER1 将创建表的权限授予 USER2。

（1）使用 WITH ADMIN OPTION 选项给 USER1 授予 CREATE TABLE 权限：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT CREATE TABLE TO USER1 WITH ADMIN OPTION;
```

（2）USER1 将 CREATE TABLE 权限授予 USER2：

```
SQL> CONN USER1/PWORDUSER1;
SQL> GRANT CREATE TABLE TO USER2;
```

（3）USER2 创建 U2T1 表：

```
SQL> CONN USER2/PWORDUSER2;
SQL> CREATE TABLE u2t1
(
    id INT,
    text VARCHAR(30)
);
```

USER2 成功创建 U2T1 表，这个例子说明了 WITH ADMIN OPTION 选项的作用。

2. 授予对象权限

1）语法格式

授予对象权限的 SQL 命令格式如下：

```
GRANT   <对象权限 1(列名)>{,<对象权限 2(列名)>}
ON <对象>
TO <用户 1>{,<用户 2>}
[WITH   GRANT   OPTION];
```

对象权限通常是 SELECT、INSERT、UPDATE、DELETE、EXECUTE、REFERENCES 等。对象通常是表、存储过程等。WITH GRANT OPTION 表示用户 1（用户 2…）获得权

限后，还可以把这个权限再次授予其他用户。既可以将整个表的某项权限授予其他用户，又可以只将表的某个字段的权限授予其他用户。

在授予对象权限时，不仅要说明是什么权限，还要指定是对哪个对象的访问权限，这是与数据库权限的授予不同的地方。

2）应用举例

【例 7-3】将一个用户的表的 SELECT 权限授予另一个用户，并可再次授权。以用户 SYSDBA 登录系统，将 DMHR 模式下的 CITY 表的 SELECT 权限授予用户 USER1。

（1）授权前用户 USER1 尝试查询 CITY 表：

```
SQL> CONN user1/PWORDUSER1;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
SELECT * FROM dmhr.city WHERE city_id='BJ';
[-5504]:没有[CITY]对象的查询权限.
```

（2）给用户 USER1 授予 CITY 表查询权限：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT SELECT ON dmhr.city TO user1 WITH GRANT OPTION;
```

（3）授权后用户 USER1 尝试查询 CITY 表：

```
SQL> CONN user1/PWORDUSER1;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
行号        CITY_ID        CITY_NAME        REGION_ID
---------- ------- --------- ----------------------------------------------------------------
1                BJ               北京                    1
```

【例 7-4】将一个用户的表的 SELECT 权限授予所有用户。以用户 SYSDBA 登录系统，将 DMHR 模式下的 LOCATION 表的 SELECT 权限授予所有用户。

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT SELECT ON dmhr.location TO PUBLIC;
SQL> CONN user1/PWORDUSER1;
SQL> SELECT * FROM dmhr.location WHERE city_id='BJ';
行号        LOCATION_ID          STREET_ADDRESS          POSTAL_CODE     CITY_ID
---------- ----------- -------------------- ----------- ----------------------------------------------------------------------------------------
1                1                  海淀区北三环西路 48 号          100086          BJ
```

【例 7-5】将一个用户的表的某个字段的 INSERT 和 UPDATE 权限授予另一个用户。以用户 SYSDBA 登录系统，将 DMHR 模式下的 CITY 表的 CITY_ID 字段的 INSERT 权限和 CITY_NAME 字段的 UPDATE 权限授予用户 USER1。

（1）授权前用户 USER1 尝试查询并修改 CITY 表数据：

```
SQL> CONN user1/PWORDUSER1;
SQL> UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
[-5503]:没有[CITY]对象的更新权限.
```

（2）授予用户 USER1 修改 CITY 表数据的权限：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT INSERT(city_id),UPDATE(city_name) ON dmhr.city TO user1;
```

（3）授权后用户 USER1 尝试修改 CITY 表的 CITY_NAME 字段值：

```
SQL> CONN user1/PWORDUSER1;
SQL> UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
SQL>COMMIT;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
行号        CITY_ID        CITY_NAME        REGION_ID
---------- ------- --------- ----------------------------------------------------------
1                BJ               北京                    1
```

这样 USER1 对 CITY 表的 CITY_ID 字段有插入权限，对 CITY_NAME 字段有修改权限，对于其他字段就没有这些权限了，这是安全控制的有效举措。对于 SELECT、INSERT 和 UPDATE 三个对象权限，还可以指定是在表中的哪个列上具有访问权限，也就是说，可以规定其他用户可以对表中的哪个列进行查询、插入和修改操作。

【例 7-6】将一个用户的存储过程的执行权限授予另一个用户。以用户 SYSDBA 登录系统，在 DMHR 模式下创建一个 SETBJNULL 存储过程并将其执行权限授予 USER1。

（1）创建 SETBJNULL 存储过程：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> CREATE OR REPLACE PROCEDURE dmhr.setbjnull AS
BEGIN
   UPDATE dmhr.city SETcity_name=NULL WHERE city_id='BJ';
   COMMIT;
END;
/
```

该存储过程将 CITY_ID 为'BJ'的 CITY_NAME 置为空。

（2）将 SETBJNULL 的执行权限授予 USER1：

```
SQL> GRANT EXECUTE ON dmhr.setbjnull TO user1;
```

（3）USER1 执行 SETBJNULL 存储过程：

```
SQL> CONN user1/PWORDUSER1;
SQL> EXECUTE dmhr.setbjnull;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
行号        CITY_ID        CITY_NAME        REGION_ID
---------- ------- --------- ----------------------------------------------------------
1                BJ               NULL                    1
```

【例 7-7】将一张表的 REFERENCES 权限授予另一个用户。以用户 SYSDBA 登录系统，将 DMHR 模式下的 CITY 表的 REFERENCES 权限授予 USER1。

（1）将 CITY 表的 REFERENCES 权限授予 USER1：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT REFERENCES(city_id) ON dmhr.city TO user1;
```

由于 city_id 是 CITY 表的主键，所以授予权限时应指定 city_id 字段。

（2）测试授权效果，使用 USER1 创建 U1TAB1 表，该表参照 DMHR.CITY 表：

```
SQL> CONN user1/PWORDUSER1;
SQL> CREATE TABLE u1tab1
(
id INT PRIMARY KEY,
cityid CHAR(2) FOREIGN KEY REFERENCES dmhr.city(city_id),
note VARCHAR(300)
);
SQL> INSERT INTO u1tab1 VALUES(1, 'BJ', '测试');
SQL> SELECT * FROM u1tab1;
行号          ID              CITYID          NOTE
---------- ---------- ------ ----------------------------------------
1             1               BJ              测试
```

【例 7-8】一个用户将获得的权限再次授予另一个用户。用户 USER1 把获得的查询 DMHR.CITY 表的权限授予用户 USER2。

（1）USER1 把获得的查询 DMHR.CITY 表的权限授予用户 USER2：

```
SQL> CONN user1/PWORDUSER1;
SQL> GRANT SELECT ON dmhr.city TO user2;
```

（2）USER2 尝试查询 DMHR.CITY 表的数据：

```
SQL> CONN user2/PWORDUSER2;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
行号          CITY_ID         CITY_NAME          REGION_ID
---------- ------- --------- ------------------------------------------------------------
1               BJ              NULL                    1
```

7.1.3　回收权限

1. 回收数据库权限

1）语法格式

回收数据库权限的 SQL 命令格式如下：

```
REVOKE   <数据库权限 1>{,<数据库权限 2>}
FROM <用户 1>{,<用户 2>}
```

这条命令一般由 SYSDBA 执行。如果一个用户在接受某个数据库权限时是以“WITH ADMIN OPTION”方式接受的，其随后又将这个数据库权限授予了其他用户，那么其也可

以将这个数据库权限从其他用户回收。

2）应用举例

【例 7-9】从用户收回用 WITH ADMIN OPTION 方式授予的权限。以用户 SYSDBA 登录系统，从用户 USER1 回收 CREATE TABLE 权限。

（1）从 USER1 回收 CREATE TABLE 权限：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> REVOKE CREATE TABLE FROM user1;
```

（2）USER1 尝试创建 U1T2 表：

```
SQL> CONN USER1/PWORDUSER1;
SQL> CREATE TABLE u1t2
(
    id INT,
    text VARCHAR(30)
);
```

USER1 创建 U1T2 表失败，因为它没有创建表的权限。

（3）USER2 尝试创建 U2T2 表：

```
SQL> CONN USER2/PWORDUSER2;
SQL> CREATE TABLE u2t2
(
    id INT,
    text VARCHAR(30)
);
```

USER2 创建 U2T2 表成功，这个例子说明，数据库权限可以转授，但是回收时不能间接回收。也就是说，SYSDBA 将某权限授予了 USER1，USER1 又将该权限授予了 USER2，当 SYSDBA 从用户 USER1 中收回该权限时，USER2 仍然拥有这个权限。

2. 回收对象权限

1）语法格式

回收对象权限的 SQL 命令格式如下：

```
REVOKE   <对象权限 1>{,<对象权限 2>} ON <对象>
FROM <用户 1>{,<用户 2>}
[RESTRICT | CASCADE]
```

回收对象权限的操作由一般权限的授予者完成。如果某个对象权限是以“WITH GRANT OPTION”方式授予用户甲的，用户甲可将这个权限再授予用户乙。从用户甲回收对象权限时，需要指定为 CASCADE，进行级联回收；如果不指定，默认为 RESTRICT，则无法进行回收。

2）应用举例

【例 7-10】从一个用户回收以 WITH GRANT OPTION 方式授予的权限。从 USER1 回

收对 DMHR 模式 CITY 表的查询权限。

（1）采用默认的 RESTRICT 模式从 USER1 回收权限，查询 CITY 表权限：

```
SQL>CONN SYSDBA/SYSDBA;
SQL> REVOKE SELECT ON DMHR.CITY FROM USER1;
REVOKE SELECT ON DMHR.CITY FROM USER1;
第 1 行附近出现错误[-5582]:回收权限无效.
```

由于先前采用 WITH GRAND OPTION 模式授予权限，所以不能以默认的 RESTRICT 模式回收权限。

（2）采用 CASCADE 模式回收权限：

```
SQL> REVOKE SELECT ON DMHR.CITY FROM USER1 CASCADE;
```

USER2 用户尝试查询 DMHR.CITY 表的数据：

```
SQL> CONN user2/PWORDUSER2;
SQL> SELECT * FROM dmhr.city WHERE city_id='BJ';
SELECT * FROM dmhr.city WHERE city_id='BJ';
[-5504]:没有[CITY]对象的查询权限.
```

这个例子说明采用级联模式从 USER1 回收查询权限后，USER2 也没有查询 CITY 表的权限。

（3）测试 USER1 的 CITY 表更新操作：

```
SQL> CONN user1/PWORDUSER1;
SQL> UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
[-5508]:没有[CITY]对象的[CITY_ID]列的查询权限.
```

由于 USER1 没有 CITY_ID 字段的查询权限，所以不能进行条件更新，只能全表更新，操作如下：

```
SQL> UPDATE dmhr.city SET city_name=NULL;
SQL> COMMIT;
SQL> CONN DMHR/DMHR12345;
SQL> SELECT COUNT(*) FROM dmhr.city WHERE city_name IS NULL;
行号        COUNT(*)
---------- --------------------
1           11
```

结果表明 USER1 对 CITY 表的更新是有效的，这个例子说明，虽然回收了 USER1 对 CITY 表的查询权限，但是由于没有回收修改数据权限，因此 USER1 还是能够对 USER1 表的数据进行修改。

为了后面内容的学习，这里把数据恢复到修改前的状态。

```
SQL> CONN DMHR/DMHR12345;
SQL> UPDATE dmhr.city SET city_name='北京' WHERE city_id='BJ';
SQL> COMMIT;
```

7.2 角色管理

角色是一组权限的组合，使用角色的目的是使权限管理更加方便。假设有 10 个用户，这些用户为了访问数据库，至少拥有 CREATE TABLE、CREATE VIEW 等权限。如果将这些权限分别授予这些用户，那么需要进行的授权次数是比较多的。但是如果把这些权限事先放在一起，然后作为一个整体授予这些用户，那么每个用户只需一次授权，授权的次数将大大减少，而且用户数越多，需要指定的权限越多，这种授权方式的优越性就越明显。这些事先组合在一起的一组权限就是角色，角色中的权限既可以是数据库权限，又可以是对象权限。

为了使用角色，首先在数据库中创建一个角色，这时角色中没有任何权限；然后向角色中添加权限；最后将这个角色授予用户，这个用户就具有了角色中的所有权限。在使用角色的过程中，可以随时向角色中添加权限，也可以随时从角色中删除权限，用户的权限也随之改变。如果要回收所有权限，只需将角色从用户回收即可。

在数据库中有两类角色，一类是 DM 预设定的角色，一类是用户自定义的角色。DM 预定义的角色在数据库被创建之后即存在，并且已经包含了一些权限，数据库管理员可以将这些角色直接授予用户。

为了保证数据库系统的安全性，达梦数据库采用“三权分立”或“四权分立”的安全机制。“三权分立”时系统内置三种系统管理员，包括数据库管理员、数据库安全员和数据库审计员，表 7-3 列出了“三权分立”常见的系统角色及其包含的部分权限。“四权分立”是新增了一类用户，称为数据库对象操作员。它们各司其职，互相制约，有效地避免了将所有权限集中于一人的风险，保证了系统的安全性。

表 7-3 数据库常见预设角色

角色名称	所包含的权限
DBA	ALTER DATABASE
	BACKUP DATABASE
	CREATE USER
	CREATE ROLE
	SELECT ANY TABLE
	CREATE ANY TABLE
RESOURCE	CREATE ROLE
	CREATE SCHEMA
	CREATE TABLE
	CREATE VIEW
	CREATE SEQUENCE
PUBLIC	SELECT TABLE
	UPDATE TABLE
	SELECT USER

（续表）

角色名称	所包含的权限
DB_AUDIT_ADMIN	CREATE USER
	AUDIT DATABASE
DB_AUDIT_OPER	AUDIT DATABASE
DB_POLICY_ADMIN	CREATE USER
	LABEL DATABASE
DB_POLICY_OPER	LABEL DATABASE

7.2.1　创建角色

除了 DM 预设定的角色以外，用户还可以自己定义角色。一般情况下创建角色的操作只能由 SYSDBA 完成，如果普通用户要定义角色，则其必须具 CREATE ROLE 数据库权限。

1. 语法格式

创建角色的 SQL 命令格式如下：

```
CREATE ROLE　角色名;
```

2. 应用举例

【例 7-11】创建角色。创建名为 ROLE1 的角色。

```
SQL> CONN SYSDBA/SYSDBA;
SQL> CREATE ROLE role1;
```

7.2.2　管理角色权限

角色刚被创建时，没有包含任何权限。用户可以将权限授予该角色，使这个角色成为一个权限的集合。

1. 授予权限

向角色授权的方法与向用户授权的方法是相同的，只要将用户名用角色名代替即可。需要注意的是，向角色授予数据库权限时，可以使用 WITH ADMIN OPTION 选项，但是向角色授予对象权限时，WITH GRANT OPTION 将无意义。

【例 7-12】给角色授予数据库权限。给 ROLE1 角色授予 ALTER USER 和 CREATE VIEW 权限。

```
SQL> GRANT ALTER USER,CREATE VIEW TO role1;
```

【例 7-13】给角色授予对象权限。给 ROLE1 角色授予查询和更新 DMHR.CITY 表的权限。

```
SQL> GRANT SELECT,UPDATE ON dmhr.city TO role1;
```

2. 回收权限

如果要从角色中删除权限，则可以执行 REVOKE 命令。权限回收的方法与从用户回收权限的方法相同，只是用角色名代替了用户名。

【例 7-14】从角色回收权限。从角色 ROLE1 回收更新 DMHR.CITY 表的权限。

```
SQL> REVOKE UPDATE ON dmhr.city FROM role1;
```

7.2.3 分配与回收角色

1. 分配角色

只有将角色授予用户，用户才会具有角色中的权限。可以一次将角色授予多个用户，这样这些用户就都具有了这个角色中包含的权限。将角色授予用户的命令是 GRANT，授予角色的方法与授予权限的方法相同，只要将权限名用角色名代替即可。在一般情况下，将角色授予用户的操作是由 DBA 用户完成的，普通用户如果要完成这样的操作，必须具有 ADMIN ANY ROLE 的数据库权限，或者具有相应的角色及其转授权。

【例 7-15】给用户分配角色。将 ROLE1 角色分配给 USER1 和 USER2 用户。

（1）分配角色前，USER2 查询 DMHR.CITY 表数据：

```
SQL> CONN USER2/PWORDUSER2;
SQL> SELECT * FROM DMHR.CITY;
SELECT * FROM DMHR.CITY;
[-5504]:没有[CITY]对象的查询权限.
```

（2）给用户 USER1 和 USER2 分配 ROLE1 角色：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> GRANT ROLE1 TO USER1,USER2;
```

（3）USER2 再次查询 DMHR.CITY 表数据：

```
SQL> CONN USER2/PWORDUSER2;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
行号        CITY_ID        CITY_NAME        REGION_ID
---------- ------- --------- ------------------------------------------------------------
1                BJ                北京                1
```

这个例子说明给用户分配角色与直接给用户授予权限具有相同的效果。

2. 回收角色

要将角色从用户回收回来，需要执行 REVOKE 命令。角色被回收后，用户所具有的属于这个角色的权限都将被回收。从用户回收角色与回收权限的方法是相同的。这样的操作一般也由 DBA 用户完成。

【例 7-16】从用户回收角色。从用户 USER2 中回收 ROLE1 角色。

（1）从用户 USER2 中回收 ROLE1 角色：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> REVOKE ROLE1 FROM USER2;
```

（2）USER2 查询 DMHR.CITY 表数据：

```
SQL> CONN USER2/PWORDUSER2;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
SELECT * FROM DMHR.CITY WHERE city_id='BJ';
[-5504]:没有[CITY]对象的查询权限.
```

这个例子说明从用户回收角色与直接从用户回收权限具有相同的效果。

7.2.4　启用与停用角色

1. 停用角色

某些时候，管理员不愿意删除一个角色，但是希望这个角色失效，此时，可以使用 SP_SET_ROLE 来设置这个角色为不可用，其语法格式如下：

```
SP_SET_ROLE('角色名', 0);
```

只有拥有 ADMIN_ANY_ROLE 权限的用户才能停用角色，并且设置后立即生效；系统预设的角色是不能设置停用的，如 DBA、PUBLIC、RESOURCE。

【例 7-17】停用 ROLE1 角色。

（1）停用角色前，用户 USER1 查询 DMHR.CITY 表数据：

```
SQL> CONN USER1/PWORDUSER1;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
行号        CITY_ID      CITY_NAME      REGION_ID
---------- ------- --------- ------------------------------------------------------------
1                BJ                北京                1
```

（2）停用角色：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> SP_SET_ROLE('ROLE1',0);
```

注意：单引号中的角色名区分大小写。

（3）停用角色后，用户 USER1 查询 DMHR.CITY 表数据：

```
SQL> CONN USER1/PWORDUSER1;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
SELECT * FROM DMHR.CITY WHERE city_id='BJ';
[-5508]:没有[CITY]对象的[CITY_ID]列的查询权限.
```

这个例子说明，当停用角色后，用户就没有了角色中的权限，无法进行相关操作了。

2. 启用角色

根据需要，可以随时启用被停用的角色，其语法格式如下：

```
SP_SET_ROLE('角色名',1);
```

只有拥有 ADMIN_ANY_ROLE 权限的用户才能停用角色，并且设置后立即生效。

【例 7-18】启用角色。启用 ROLE1 角色。

（1）启用 ROLE1 角色：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> SP_SET_ROLE('ROLE1',1);
```

（2）启用角色后，用户 USER1 查询 DMHR.CITY 表数据：

```
SQL> CONN USER1/PWORDUSER1;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
行号        CITY_ID        CITY_NAME        REGION_ID
---------- ------- --------- ------------------------------------------------------------
1                BJ                北京                1
```

这个例子说明当启用角色后，用户立即具有了角色中的权限。

7.2.5 删除角色

有时需要彻底删除角色而不是停用，角色被删除时，角色中的权限都间接地被从用户中回收，效果与停用角色相同。删除角色的语法格式如下：

```
DROP ROLE 角色名;
```

【例 7-19】删除 ROLE1 角色。

（1）删除角色：

```
SQL> CONN SYSDBA/SYSDBA;
SQL> DROP ROLE role1;
```

（2）删除角色后，用户 USER1 查询 DMHR.CITY 表数据：

```
SQL> CONN USER1/PWORDUSER1;
SQL> SELECT * FROM DMHR.CITY WHERE city_id='BJ';
SELECT * FROM DMHR.CITY WHERE city_id='BJ';
[-5508]:没有[CITY]对象的[CITY_ID]列的查询权限.
```

这个例子说明，删除角色后，用户依赖于该角色的权限就没有了。

7.3 数据库审计

审计机制是达梦数据库管理系统安全管理的重要组成部分之一。达梦数据库除了提供数据安全保护措施之外，还提供对日常事件的事后审计监督。DM 具有一个灵活的审计子系统，可以通过它来记录系统级事件、个别用户的行为以及对数据库对象的访问。通过考

察、跟踪审计信息，数据库审计员可以查看用户访问数据库的形式以及曾试图对该系统进行的操作，从而采取积极有效的应对措施。

DM 中设立了 AUDITOR（审计员）角色，只有具有 AUDITOR 角色权限的用户才能进行审计，可以通过对用户授予 AUDITOR 角色来给予某个用户审计的权限。只有数据库审计管理员才能创建新的审计员。

一个数据库审计员能够回收另一个数据库审计员的审计权限，但由系统最初定义的数据库审计员 SYSAUDITOR 的权限是不能被回收的。

7.3.1　设置数据库审计

数据库审计员指定被审计对象的活动称为审计设置。DM 提供了审计设置系统函数来实现这种设置。

1. 启用审计功能

在 DM 系统中，专门为审计设置了开关，要使用审计功能首先要打开审计开关，开关打开后，DM 对审计设置语句指定的审计对象进行审计，否则不记录审计记录。

1）打开审计开关

此开关的设置方法如下：通过在控制台中配置服务器 ENABLE_AUDIT 参数值启用该功能，设置为 1 表示打开普通审计，设置为 2 表示打开普通审计和实时审计，设置为 0 则表示关闭审计开关，此配置项默认值为 0。为了实时观察审计效果，这里将 ENABLE_AUDIT 设置为 2，如图 7-1 所示。

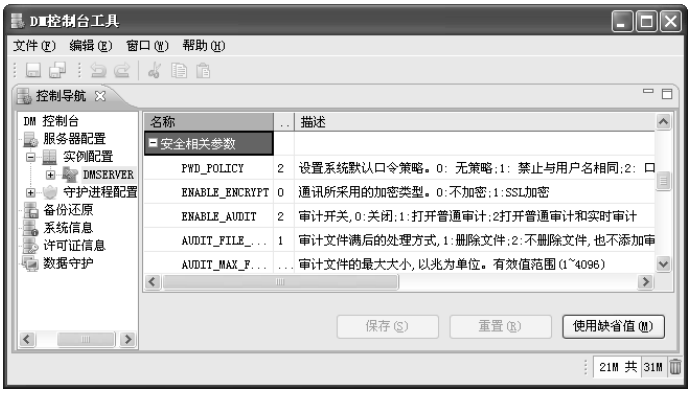

图 7-1　设置 ENABLE_AUDIT 参数值

2）启用审计功能

打开审计开关后，需要重新启动 DM 服务器方可生效。在 DM 服务查看器中单击“重启动”按钮即可，如图 7-2 所示，DM 自动在 C:\dmdbms\data\DAMENG 目录下生成一个名为 AUDIT_94C0A08005C94C AD9129815B23B0E201_2016-5-16-11-22-20.log（一个具有唯一性的文件名）的审计日志文件。

2. 设置审计范围

被审计的对象可以是某类操作，也可以是某些用户在数据库中的全部行踪。只有预先设置的操作和用户才能被 DM 系统自动进行审计。当数据库审计员认为某些操作或某些用户不必进行审计时，可将原来对某些操作或用户的审计设置清除。设置与取消审计是立即生效的，不需重新启动服务器。

审计设置存放于系统表中，进行一次审计设置会在系统表 SYSAUDIT 中增加设置的记录，取消审计则删除系统表 SYSAUDIT 中相应的记录。

图 7-2　重启达梦数据库服务

达梦数据库审计分为以下三个级别。

（1）系统级，即系统的启动与关闭，此级别的审计记录在任何情况下都会强制产生，无法也无需由用户进行设置。

（2）语句级，即导致影响特定类型数据库对象的特殊 SQL 或语句组的审计。例如，ADUIT TABLE 将审计 CREATE TABLE、ALTER TABLE 和 DROP TABLE 等语句。语句级审计的动作是全局的，不对应具体的数据库对象。语句级审计选项摘要介绍见表 7-4，详细内容见《DM 7_DBA》电子版。

表 7-4　语句级审计选项

审计选项	审计的数据库操作	说　　明
ALL	所有的语句级审计选项	所有可审计操作
USER	CREATE USER ALTER USER DROP USER	创建/修改/删除用户操作
ROLE	CREATE ROLE DROP ROLE	创建/删除角色操作

（3）对象级，即审计作用在特殊对象上的语句，如 CITY 表上的 INSERT 语句。对象级审计发生在具体的对象上，需要指定模式名以及对象名。对象级审计选项见表 7-5。

表 7-5　对象级审计选项

审计选项（SYSAUDITRECORDS 表中 operation 字段对应的内容）	表	视图	PROCEDURE FUNCTION
INSERT（insert）	√	√	
UPDATE（update）	√	√	
DELETE（delete）	√	√	
SELECT（select）	√	√	
EXECUTE（execute）			√
MERGE INTO	√	√	
EXECUTE TRIGGER			
LOCK TABLE	√		
ALL（所有对象级审计选项）	√	√	√

3. 语法格式

（1）设置语句级审计选项：

```
SP_AUDIT_STMT('语句级审计选项','用户名','审计时机');
```

语句级审计选项参见表 7-4，审计时机包括 ALL/SUCCESSFUL/FAIL 三种，对应全部审计/成功时审计/失败时审计。如果用户操作了数据，并且符合审计时机条件，则数据库会自动记录用户的操作用于审计。

（2）取消语句级审计选项：

```
SP_NOAUDIT_STMT('语句级审计选项','用户名','审计时机');
```

（3）设置对象级审计选项：

```
SP_AUDIT_OBJECT('对象级审计选项','用户名','模式名','对象名','审计时机');
```

对象级审计选项见表 7-5，审计时机包括 ALL/SUCCESSFUL/FAIL 三种，对应全部审计/成功时审计/失败时审计。

（4）取消对象级审计选项：

```
SP_NOAUDIT_OBJECT('对象级审计选项','用户名','模式名','对象名','审计时机');
```

4. 应用举例

【例 7-20】设置语句级审计。对用户 DMHR 修改表数据时进行审计，不管失败和成功。

（1）设置语句级审计。

```
SQL> CONN SYSAUDITOR/SYSAUDITOR;
SQL> SP_AUDIT_STMT('UPDATE TABLE','DMHR','ALL');
```

（2）执行语句级操作。

```
SQL> CONN DMHR/DMHR12345;
SQL> UPDATE CITY SET CITY_NAME='北京' WHERE CITY_ID='BJ';
```

```
SQL> COMMIT;
```

（3）查看审计结果：审计结果如图 7-3 所示，第一条数据是数据库重启，记录的是图 7-2 中重启数据库的审计结果，它是系统级审计；第二条数据就是（2）中 DMHR 用户更新 CITY 表数据的审计结果。

图 7-3　数据库审计结果

【例 7-21】取消语句级审计。取消对用户 DMHR 修改表数据失败时的审计。

```
SQL> SP_NOAUDIT_STMT('UPDATE TABLE','DMHR','FAIL');
SP_NOAUDIT_STMT('UPDATE TABLE','DMHR','FAIL');
[-5303]:此审计类型未设置.
```

取消语句级审计失败，原因是此前并没有设置过该语句级审计，因此不存在取消审计。这个例子说明，即使在例 7-20 中设置'UPDATE TABLE'的审计时机为'ALL'，并且 ALL 包括'SUCCESSFUL'和'FAIL'两种时机，但是仍然不能单独取消 FAIL 时机的审计。

【例 7-22】取消语句级审计。取消对用户 DMHR 修改表数据失败时的审计。

（1）取消语句级审计。

```
SQL> CONN SYSAUDITOR/ SYSAUDITOR;
SQL> SP_NOAUDIT_STMT('UPDATE TABLE','DMHR','ALL');
```

（2）执行语句级操作。

```
SQL> CONN DMHR/DMHR12345;
SQL> UPDATE CITY SET CITY_NAME='中国北京' WHERE CITY_ID='BJ';
SQL> COMMIT;
```

（3）查看审计结果；审计结果如图 7-3 所示，并没有增加新的审计记录。这个例子说明，如果取消了语句级审计，后续的语句级的操作就不会被记录下来。这个例子还说明，已经记录下来的审计结果会一直保留下来，不会因为取消审计而删除。

【例 7-23】设置对象级审计，成功时审计。设置对用户 DMHR 向 CITY 表插入数据成功时的审计。

（1）设置对象级审计，成功时审计。

```
SQL> CONN SYSAUDITOR/ SYSAUDITOR;
SQL> SP_AUDIT_OBJECT('INSERT','DMHR','DMHR','CITY','SUCCESSFUL');
```

（2）执行对象级操作，包括成功和失败操作。

```
SQL> CONN DMHR/DMHR12345;
SQL> INSERT INTO CITY(CITY_ID, CITY_NAME) VALUES('BA','博鳌');
```

```
SQL> INSERT INTO CITY(CITY_ID, CITY_NAME) VALUES('BJ','苏州');
SQL> COMMIT;
```

执行上述操作时，插入第一条数据“博鳌”成功，插入第二条数据“苏州”失败，原因是第二条数据的主键'BJ'出现了重复。

（3）查看审计结果；审计结果如图 7-4 所示，插入“博鳌”数据的操作被记录下来，插入“苏州”数据的操作没有记录。这是因为设置对象级审计时，只设置成功时审计，失败时不记录。

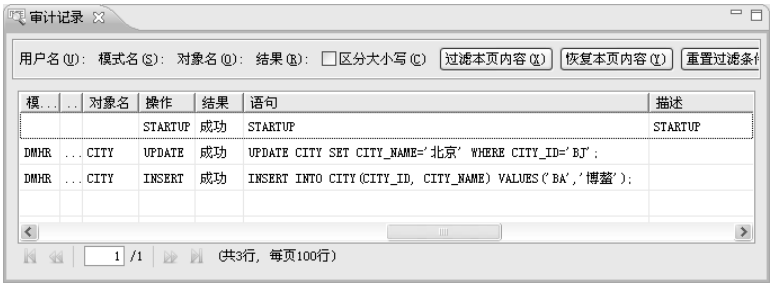

图 7-4　数据库审计结果

【例 7-24】设置对象级审计，失败时审计。设置对用户 DMHR 删除 CITY 表数据失败时的审计。

（1）设置对象级审计，失败时审计。

```
SQL> CONN SYSAUDITOR/SYSAUDITOR;
SQL> SP_AUDIT_OBJECT('DELETE','DMHR','DMHR','CITY','FAIL');
```

（2）执行对象级操作，包含失败操作。

```
SQL> CONN DMHR/DMHR12345;
SQL> DELETE FROM CITY WHERE CITY_ID='BA';
SQL> DELETE FROM CITY WHERE CITY_NAME='苏州';
SQL> DELETE FROM CITY WHERE CITYNAME='北京';
SQL> DELETE FROM CITY WHERE CITY_ID='BJ';
SQL> COMMIT;
```

上述语句执行时，第一条删除数据操作成功；第二条删除数据操作虽然没有删除数据，但是命令本身是成功的；第三条删除数据操作失败，没有 CITYNAME 字段；第四条删除数据操作失败，违反参照完整性约束，LOCATION 表的 CITY_ID 外键字段引用了 CITY 表的 CITY_ID 为'BJ'的值。

（3）查看审计结果；审计结果如图 7-5 所示，两条失败的操作都被记录下来，无论是语法错误，还是完整性错误，只要是失败的操作都会被记录下来。

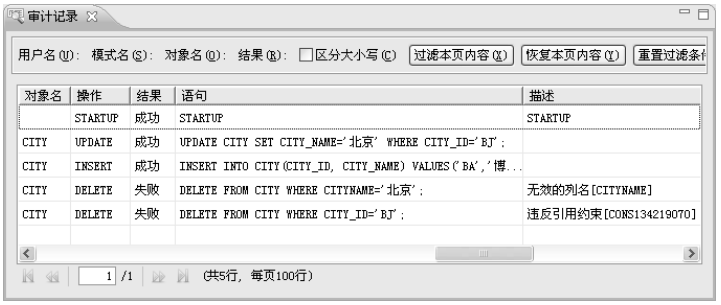

图 7-5　数据库审计结果

【例 7-25】取消对象级审计。取消对用户 DMHR 删除 CITY 表数据失败时的审计。

```
SQL> CONN SYSAUDITOR/ SYSAUDITOR ;
SQL> SP_NOAUDIT_OBJECT('DELETE','DMHR','DMHR','CITY','FAIL');
```

在这个例子中，取消删除表数据失败时审计是指新删除数据失败的操作不会产生审计记录，但是过去的删除数据失败的审计记录仍然存在。

5. 附加说明

尽管相对来说审计的开销并不是很大，但是要尽可能地限制审计对象和审计事件的数目，最大限度地降低审计带来的性能方面的影响，并且减小审计记录的规模。当使用审计的时候，通常考虑以下问题。

（1）评价审计的目的。在对审计的目的和原因有了清晰的认识后，可以设计合适的审计策略来避免不必要的审计。

（2）理智地进行审计。审计必要的最少数目的语句、用户或者对象来获得目标信息。这样，避免在混乱的审计信息中浪费精力。

（3）审计操作的甄选。在开始审计的时候，也许不清楚应该审计哪些可疑操作，这时可以适当扩大审计操作的范围，一旦记录并分析了初步的审计信息，就应该取消一些普通的操作，而把审计的重点放在一些可疑操作上。

（4）存档审计并清除审计跟踪。在收集到必要的审计信息后，应该通过存档审计文件来保留这些审计记录，并清除这些信息的审计跟踪。

（5）当使用 DM 提供的审计机制进行了审计设置后，这些审计设置信息都记录在数据字典中。只要 DM 系统处于审计活动状态，系统将按审计设置进行审计活动，并将审计信息写入审计文件。审计记录内容包括操作者的用户名、所在站点、所进行的操作、操作的对象、操作时间、当前审计条件等。具有审计权限的用户可以通过系统表 SYSAUDIT 查询审计设置信息，通过动态视图 V$AUDITRECORDS 查询系统默认路径下的审计文件的审计记录。

7.3.2　分析审计结果

DM 提供了专门的审计分析工具 Analyzer，实现对审计记录的分析功能，能够根据所

制定的分析规则，对审计记录进行分析，判断系统中是否存在对系统安全构成危险的活动。只有审计用户才能使用审计分析工具。审计用户登录审计分析工具后，通过 Analyzer 可以创建和删除审计规则，并可以指定对某些审计文件应用某些规则，将审计结果以表格的方式展现出来。

审计分析规则由一系列的条件组合而成，这些条件包括：用户名、模式名、审计对象名、审计操作名、操作时间段、IP 段、时间间隔和门限次数以及这些条件的组合。对审计文件进行分析时，可以同时应用多个规则，分析结果为应用各个分析规则的交集。

1. 直接查看审计结果

审计信息存储在审计文件中。审计文件命名格式为“AUDIG_GUID_创建时间.log”。当单个审计文件超过指定大小时，系统会自动切换审计文件，自动创建新的审计文件，审计记录将写入新的审计文件中。审计文件的大小可以通过 INI 参数 AUDIT_MAX_FILE_SIZE 指定。随着系统的运行，审计记录将会不断增加，审计文件需要更多的磁盘空间。在极限情况下，审计记录可能会因为磁盘空间不足而无法写入审计文件，最终导致系统无法正常运行。DM 采用了两种策略处理此情况：一种是配置 INI 参数 AUDIT_FILE_FULL_MODE＝1，删除最旧的审计文件，直至有足够的空间创建新审计文件；另一种是配置 INI 参数 AUDIT_FILE_FULL_MODE＝2，将不再写审计记录。此参数可通 SP_SET_PARA_VALUE 动态修改。DM 默认采用第一种策略。这两种策略都会导致审计记录的缺失，因此，数据管理员应该及时对审计文件进行备份。

本例中的审计结果在 AUDIT_94C0A08005C94CAD9129815B 23B0E201_2016-5-16-11-22-20.log 文件中，并且不能直接查看这个文件，只能通过达梦数据库的 Analyzer 来查看审计结果。

【例 7-26】使用达梦数据库 Analyzer 查看审计结果。

步骤 1：运行 DM 审计分析系统工具时，必须以审计员或审计管理员的身份登录，这里使用 SYSAUDITOR 用户、密码 SYSAUDITOR 登录，成功后进入审计分析工具主界面，如图 7-6 所示。

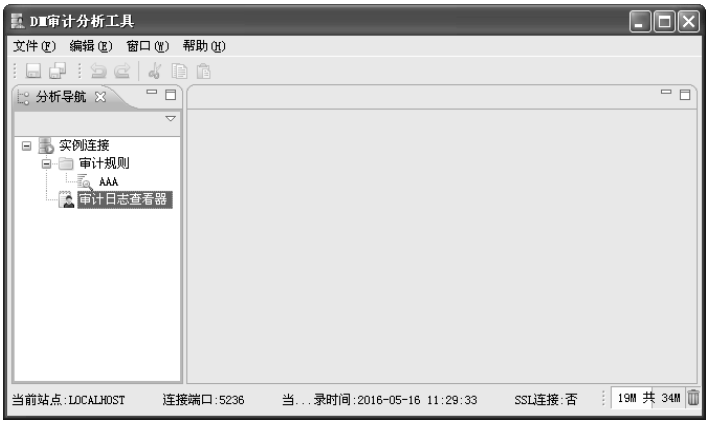

图 7-6　DM 审计分析工具主界面

步骤 2：在图 7-6 中，双击“审计日志查看器”节点，弹出“条件设置”对话框，在“日志文件”页面中单击“添加文件”按钮，将 AUDIT_94C0A08005C 94CAD9129815B23B0E201_2016-5-16-11-22-20.log 文件加入到文件列表中，如图 7-7 所示，单击“确定”按钮。

步骤 3：打开审计文件后，审计结果如图 7-8 所示，可以看到所有的审计结果，包括系统级、语句级和对象级的审计记录。

图 7-7　打开审计文件

步骤 4：当操作次数很多时，图 7-8 中的审计结果会很多，而用户可能只关心部分审计记录，这时可以设置一些过滤条件，过滤无关的审计记录，只显示关心的记录。过滤的条件包括用户名、模式名、对象名、结果、区分大小写等。其中，结果过滤项包括全部/成功/失败三种情况，将结果设置为“失败”，单击“过滤本页内容”按钮，如图 7-9 所示，原来显示 5 条审计记录，现在只显示操作“失败”的两条记录。

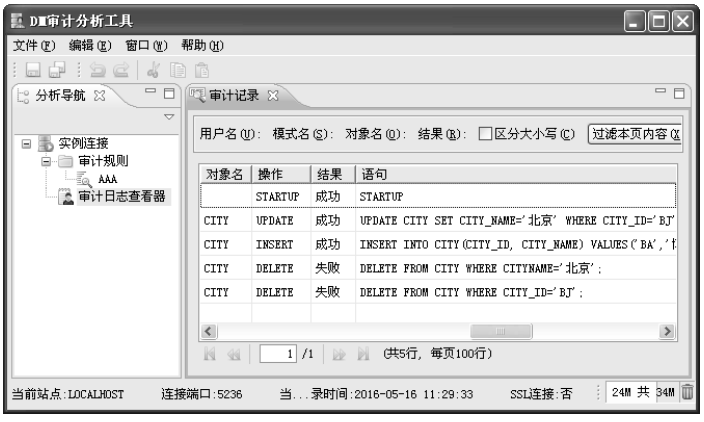

图 7-8　列表显示审计结果

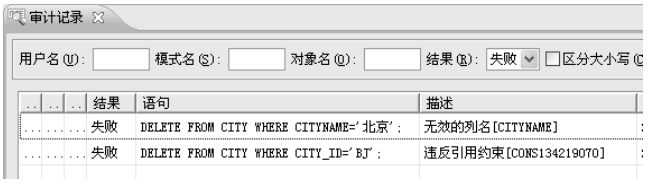

图 7-9　过滤分析审计结果

2. 按审计规则分析审计日志

按审计规则分析审计日志是在审计日志的基础上，按照指定的规则从审计日志中把关注的审计记录挑选出来。

【例 7-27】新建审计规则 DMHRDEL，分析 DMHR 用户的所有删除数据的审计记录。

步骤 1：在 DM 审计分析工具中，右击“审计规则”节点，在弹出的快捷菜单中选择“新建审计分析规则”选项，弹出“新建审计分析规则”对话框，如图 7-11 所示。

步骤 2：在图 7-11 中，规则名设置为“DMHRDEL”，在允许的用户下，单击“添加”按钮添加 DMHR 用户。

图 7-10　新建审计分析规则

图 7-11　设置常规参数

步骤 3：进入“操作类型-DML”页面，如图 7-12 所示，在操作列表中选中“DELETE”复选框，操作时机为“ALL”，单击“确定”按钮，完成 DMHRDEL 审计规则的创建。

图 7-12　设置 DML 操作类型参数

步骤 4：在审计规则树下选择 DMHRDEL 审计规则并右击，在弹出的快捷菜单中选择“审计规则分析”选项，弹出“审计规则分析”对话框，如图 7-13 和图 7-14 所示。在规则列表中已经选择 DMHRDEL 审计规则，单击“添加文件”按钮，把审计日志文件添加到文件列表中。单击“确定”按钮，回到审计分析工具主界面，显示按审计规则分析审计的结果，如图 7-15 所示。

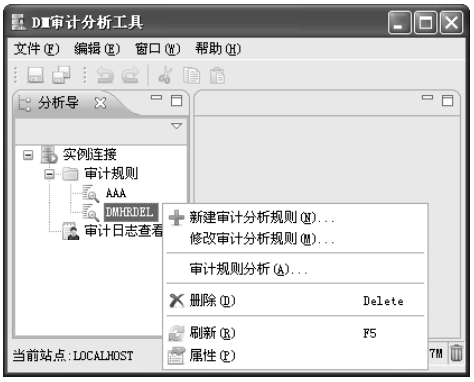

图 7-13　审计规则分析

图 7-14　选择被分析的审计日志文件

图 7-15　按审计规则分析审计的结果

步骤 5：在图 7-15 中，显示了 DMHR 用户删除表数据操作的两条审计记录，不显示其他审计记录。当同时使用多条审计规则分析一个审计日志时，显示的结果是同时满足这些审计规则的审计日志记录。

这个例子说明，按审计规则分析与过滤分析审计结果相比，虽然设置复杂了一些，但是可以设置更加精细的分析条件，分析效果更好。

7.3.3　监测实时侵害

当 INI 参数 ENABLE_AUDIT=2 时，开启审计实时侵害检测功能。

实时侵害检测系统用于实时分析当前用户的操作，并查找与该操作相匹配的实时审计分析规则，如果规则存在，则判断该用户的行为是否侵害行为，确定侵害等级，并根据侵害等级采取相应的响应措施。

1. 侵害等级

根据侵害行为对系统造成的危害程度，实时侵害检测系统定义了四种级别的侵害等级，四种侵害等级分别如下。

（1）四级：操作只违反了 IP 或者时间段设置之一，且对应累计次数在规定时间间隔内没有达到门限值，或者没有门限值设置。

（2）三级：操作同时违反了 IP 和时间段设置，且累计次数在规定时间间隔内没有达到门限值，或者没有门限值设置。

（3）二级：操作的 IP 和时间段都正常，且累计次数在规定时间间隔内达到了门限值。

（4）一级：操作只违反了 IP 或时间段设置之一，且对应累计次数在规定时间间隔内达到了门限值，或者操作同时违反了 IP 和时间段设置，且累计次数在规定时间间隔内达到了门限值。

实时侵害检测规则的设置，使得某些操作可能会触发多条审计分析规则，也就是说，可能存在多条审计分析规则同时匹配的情况。在这种情况下，需要把操作记录保存到所有匹配的、同时规定了操作频率门限值的审计分析规则下，并且使用这些审计分析规则对该操作进行分析，从所有满足的侵害等级中选择一个最高的侵害等级进行响应。

匹配实时审计分析规则按照最大匹配原则进行，即尽可能多地匹配可匹配的分析规则。判断的依据包括：登录/用户名、审计选项。对于规则中的 WHENEVER 子句，在匹配规则时可以忽略，都视为匹配，或称为不完全匹配。

2. 响应等级

为了对侵害行为及时采取措施，DM 数据库自动采取安全响应，安全响应的分类与侵害等级的分级相对应，由低到高分为四个等级。

（1）四级响应：实时报警生成，对四级侵害行为进行响应。当系统检测到四级侵害行为时，生成报警信息。

（2）三级响应：违例进程终止，对三级侵害行为进行响应。当系统检测到三级侵害行为时，终止当前操作（用户当前连接仍然保持），生成报警信息。

（3）二级响应：服务取消，对二级侵害行为进行响应。当系统检测到二级侵害行为时，强制断开用户当前连接，退出登录，生成报警信息。

（4）一级响应：账号锁定或失效，对一级侵害行为进行响应。当系统检测到一级侵害行为时，强制断开用户当前连接，退出登录，并且锁定账号或使帐号失效，生成报警信息。

3. 入侵审计

具有 AUDIT DATABASE 权限的用户可以使用下面的系统过程创建实时侵害检测规则。

```
SP_CREATE_AUDIT_RULE(
RULENAME    VARCHAR(128),
OPERATION    VARCHAR(30),
USERNAME    VARCHAR(128),
SCHNAME    VARCHAR(128),
OBJNAME    VARCHAR(128),
WHENEVER    VARCHAR(20),
ALLOW_IP    VARCHAR(1024),
ALLOW_DT    VARCHAR(1024),
INTERVAL    INTEGER,
TIMES       INTERGER
);
```

参数说明：

（1）RULENAME：创建的审计实时侵害检测规则名。

（2）OPERATION：被审计操作名，包括：CREATE USER、DROP USER、ALTER USER、CREATE ROLE、DROP ROLE、CREATE TABLESPACE、DROP TABLESPACE、ALTER_TABLESPACE、CREATE SCHEMA、DROP SCHEMA、SET SCHEMA、CREATE TABLE、DROP TABLE、TRUNCATE TABLE、CREATE VIEW、ALTER VIEW、DROP VIEW、ALTER VIEW、CREATE INDEX、DROP INDEX、CREATE PROCEDURE、ALTER PROCEDURE、DROP PROCEDURE、CREATE TRIGGER、DROP TRIGGER、ALTER TRIGGER、CREATE SEQUENCE、DROP SEQUENCE、CREATE CONTEXT INDEX、DROP CONTEXT INDEX、ALTER CONTEXT INDEX、GRANT、REVOKE、AUDIT、NOAUDIT、CHECKPOINT、CREATE POLICY、DROP POLICY、ALTER POLICY、CREATE LEVEL、ALTER LEVEL、DROP LEVEL、CREATE COMPARTMENT、DROP COMPARTMENT、ALTER COMPARTMENT、CREATE GROUP、DROP GROUP、ALTER GROUP、ALTER GROUP PARENT、CREATE LABEL、DROP LABEL、ALTER LABEL、REMOVE TABLE POLICY、APPLY TABLE POLICY、USER SET LEVELS、USER SET COMPARTMENTS、USER SET GROUPS、USER SET PRIVS、USER REMOVE POLICY、SESSION LABEL、SESSION ROW LABEL、RESTORE DEFAULT、LABELS、SAVE DEFAULT LABELS、SET TRX ISOLATION、SET TRX READ WRITE、CREATE PACKAGE、DROP PACKAGE、CREATE PACKAGE BODY、DROP PACKAGE BODY、CREATE SYNONYM、DROP SYNONYM、INSERT、SELECT、DELETE、UPDATE、EXECUTE、EXECUTE TRIGGER、MERGE INTO、LOCK TABLE、UNLOCK USER、ALTER DATABASE、SAVEPOINT、COMMIT、ROLLBACK、CONNECT、DISCONNECT、EXPLAIN、STARTUP、SHUTDOWN。

（3）USERNAME：用户名，没有指定或'NULL'表示所有用户。

（4）SCHNAME：模式名，没有时指定为'NULL'。

（5）OBJNAME：对象名，没有时指定为'NULL'。
（6）WHENEVER：审计时机，取值 ALL/SUCCESSFUL/FAIL。
（7）ALLOW_IP：IP 列表，以’,’隔开。例如'"192.168.0.1","127.0.0.1"'。
（8）ALLOW_DT：时间串，格式如下：
（9）ALLOW_DT::=　<时间段项>{,<时间段项>}
① <时间段项> ::= <具体时间段> | <规则时间段>
② <具体时间段> ::= <具体日期> <具体时间> TO <具体日期> <具体时间>
③ <规则时间段> ::= <规则时间标志> <具体时间> TO <规则时间标志> <具体时间>
④ <规则时间标志> ::= MON | TUE | WED | THURS | FRI | SAT | SUN。
（10）INTERVAL：时间间隔，单位为分钟。
（11）TIMES：次数。

4. 监测手段

达梦提供了 dmamon 工具用于监视审计实时侵害检测，并将报警信息以邮件的方式发送到邮箱中，dmamon 需要配置 dmamon.ini 文件。

1）配置参数文件 dmamon.ini

dmamon.ini 文件不能手动编辑，只能使用 dmamon_ctl 工具创建和修改，格式如下：

```
dmamon_ctl [FILE=含路径的 dmamon.ini 文件]
```

如果指定 FILE 参数，则打开参数配置文件，否则新建一个参数配置文件。

dmamon_ctl 工具在设置参数时可以使用内部命令，见表 7-6。

表 7-6　dmamon_ctl 内部命令

命　令	功　能
EDIT	修改参数文件信息
ADD	增加接收者的邮件地址信息
REMOVE	删除接收者的邮件地址信息
SHOW	显示当前控制文件的内容
SAVE	保存，路径必须为 dmamon.ini 的绝对路径。退出程序前必须保存，否则修改无效
EXIT	退出当前程序
HELP	显示帮助信息

使用 dmamon_ctl 配置 dmamon.ini 文件的参数内容见表 7-7。

表 7-7　dmamon.ini 参数文件的内容

项　目	参　数	用　途
[PATH]	PATH	实时审计记录文件所在的目录
[SENDER]	EMAIL	发件人邮箱
	SMTP	发件人邮箱 SMTP 服务器
	USERNAME	发件人用户名
	PASSWORD	发件人密码

（续表）

项　目	参　数	用　　途
[RECEIVER]	EMAIL	收件人邮箱
[INTERVAL]	INTERVAL	检查扫描实时审计记录文件的时间间隔

2）启动实时监测

dmamon 工具用于监视审计实时侵害，命令格式如下：

```
dmamon PATH=参数文件 USERID=数据库登录信息
```

参数文件是含有路径的 dmamon.ini 文件。

数据库登录信息包含用户名、密码、服务器地址、端口号，格式如下：

```
用户名/密码[@主机地址[:端口号] [#SSL_PATH@SSL_PWD]
```

5．应用举例

【例 7-28】对所有针对 DMHR 模式下 CITY 表的查询操作进行监测，当 10 秒中内进行了 3 次查询操作时立即通过邮件报警。在参数文件配置时，路径为 C:\dmdbms\data\DAMENG，报警发送邮箱为 alarm@181.com，邮箱用户名为 alarm，邮箱密码为 PWDwh123，邮件服务器地址为 192.168.106.133，报警信息接收邮箱为 water@181.com，报警扫描间隔为 10 秒。

步骤 1：如果 DM 数据库服务器部署在互联网计算机上，则可以在互联网上注册两个邮箱用于本例，如果 DM 数据库服务器部署在局域网计算机上，则应当部署邮件服务器，并注册两个邮箱，如 alarm@181.com 和 water@181.com。

步骤 2：创建入侵审计规则。

```
SQL> CONN SYSAUDITOR/SYSAUDITOR;
SQL> SP_CREATE_AUDIT_RULE('SELECTLOG', 'SELECT', 'NULL', 'DMHR', 'CITY', 'ALL', 'NULL',
'NULL',10,3);
```

步骤 3：配置参数文件。执行 dmamon_ctl 来配置 dmamon.ini 参数文件，命令如下[①]。

```
C:\dmdbms\bin>dmamon_ctl
Open dmamon.ini file failed!
错误的文件
重新生成新的文件
格式: dmamon_ctl   [FILE=value]
例程: dmamon_ctl   FILE=c:\dmamon.ini
关键字                    说明（默认）
------------------------------------------------------------------------------
edit                     修改 INI 文件信息
add                       增加接收者的邮件地址信息
remove                    删除接收者的邮件地址信息
```

[①] 此命令是在交互方式下运行，加粗文字表示手工输入内容，其他文字为系统提示内容。

```
show                    显示当前控制文件的内容
save                    保存输入信息为 ini，必须为 ini 绝对路径
exit                    退出当前程序
help                    显示帮助信息
PATH    :c:\dmdbms\data\DAMENG
sender email    :alarm@181.com
sender smtp    :192.168.106.133
sender username    :alarm
sender password    :在这里输入邮箱密码
receiver email server use ssl (y/n 1/0):n
add receiver email :water@181.com
interval    :10
[PATH]
PATH    = c:\dmdbms\data\DAMENG
[SENDER]
EMAIL    = alarm@181.com
SMTP    = 192.168.106.133
USERNAME    = alarm
PASSWORD    = ****
USE_SSL = FALSE
[RECEIVER]
EMAIL    = water@181.com
[INTERVAL]
INTERVAL    = 10
dmamon>save
save ini file to this path:
c:\dmdbms\data\DAMENG\dmamon.ini
dmamon>exit
```

步骤 4：启动侵害实时监测。启动 dmamon 来监测实时侵害，命令如下。

```
C:\dmdbms\bin>dmamon    path=c:\dmdbms\data\DAMENG\dmamon.ini    userid=SYSAUDITOR/
SYSAUDITOR@192.168.106.133
```

输入上述命令后，系统显示监测状态如下。

```
initial monitor system ...
SYSTEM IS READY.
```

步骤 5：尝试在 10 秒中内连续 3 次查询 DMHR 模式下的 CITY 表。

```
SQL> CONN dmhr/DMHR12345;
SQL> select * from city;
SQL> select * from city;
```

```
SQL> select * from city;
```

上述命令执行完毕后，触发审计规则，数据库自动采取相应级别响应措施，断开连接并且发送报警邮件。

步骤 6：进入 water@181.com 邮箱查看报警邮件，如图 7-16 所示。

图 7-16　报警邮件

邮件内容含义如下：DMHR 用户在 IP 地址为 127.0.0.1（DM 服务器本机）的计算机上，于 2016 年 5 月 24 日下午 5 时 58 分 16 秒进行了 select * from city 操作，触发了 SELECTLOG 审计规则，本次报警等级为 2 级。

附录 A　数据库参数配置

达梦数据库服务参数配置主要利用 dm.ini 参数配置文件实现。该文件在创建每个达梦数据库时由系统自动生成，主要包括控制文件、实例、内存、线程等参数。

附录中参数属性的“静态”指修改后重启服务器才能生效；“动态”指修改后即时生效；“手动”表示服务器运行过程中不可修改。动态参数又分为会话级和系统级两个级别，会话级参数在服务器运行过程中被修改时，之前创建的会话不受影响，只有新创建的会话使用新的参数值；系统级参数的修改则会影响所有的会话。

1. 控制文件参数

控制文件常见参数及含义见表 A-1。

表 A-1　控制文件参数及含义

参数名	默认值	属性	说　明
CTL_PATH	安装时指定	手动	控制文件路径
TEMP_PATH	安装时指定	手动	临时库文件路径
BAK_PATH	安装时指定	手动	备份路径

2. 内存参数

内存常见参数及含义见表 A-2。

表 A-2　内存常见参数及含义

参 数 名	默认值	属性	说　明
MAX_OS_MEMORY	90	静态	DM 服务器能使用的最大内存占操作系统物理内存与虚拟内存总和的百分比，有效值为 40～100。当取值 100 时，服务器不进行内存的检查。注：对于 32 位版本的 DM 服务器而言，虚拟内存最大为 2GB
MEMORY_POOL	40	静态	系统内存池大小，以 MB 为单位。系统内存池是由 DM 管理的内存，系统内存的申请都从内存池中申请。有效值为 5～10000

（续表）

参 数 名	默认值	属性	说　明
MEMORY_BAK_POOL	4	静态	系统备份内存池大小，以 MB 为单位。系统备份内存池是由 DM 管理的内存。有效值为 2～10000
HUGE_BUFFER	8	静态	HFS 表使用的缓冲区大小，以 MB 为单位。有效值为 8～1048576
BUFFER	100	静态	系统缓冲区大小，以 MB 为单位。推荐值：系统缓冲区大小为可用物理内存的 60%～80%。有效值为 8～1048576
BUFFER_POOLS	1	静态	BUFFER 系统分区数（1～512）
FAST_POOL_PAGES	0	静态	快速缓冲区页面数（0～99999）
FAST_ROLL_PAGES	0	静态	快速缓冲区回滚页面数（0～99999）
KEEP	8	静态	KEEP 缓冲区大小，以 MB 为单位。有效值为 8～1048576
RECYCLE	64	静态	RECYCLE 缓冲区大小，以 MB 为单位。有效值为 8～1048576
MULTI_PAGE_GET_NUM	16	动态，系统级	缓冲区最多一次读取的页面数。有效值为 1～128
MAX_BUFFER	100	静态	系统最大缓冲区大小，以 MB 为单位。采用动态缓冲区时，可以扩展到的最大缓冲区页数（>=BUFFER）。有效值为 8～1048576
SORT_BUF_SIZE	2	动态，会话级	排序缓存区大小，以 MB 为单位。有效值为 1～1024
HAGR_HASH_SIZE	100000	动态，会话级	HAGR 操作时，建立 HASH 表的个数。有效值为 10000～100000000
HJ_BUF_GLOBAL_SIZE	500	手动	HASH 连接操作符的数据总缓存大小（>=HJ_BUF_SIZE），系统级参数，以 MB 为单位。有效值为 10～500000
HJ_BUF_SIZE	50	动态，会话级	单个 HASH 连接操作符的数据总缓存大小，以 MB 为单位。有效值为 2～100000
HJ_BLK_SIZE	1	动态，会话级	HASH 连接操作符每次分配缓存（BLK）大小，以 MB 为单位，必须小于 HJ_BUF_SIZE。有效值为 1～50
HAGR_BUF_GLOBAL_SIZE	500	手动	HAGR、DIST、集合操作、SPOOL、NTTS 以及 HTAB 操作符的数据总缓存大小（>=HAGR_BUF_SIZE），系统级参数，以 MB 为单位。有效值为 10～1000000
HAGR_BUF_SIZE	50	动态，会话级	单个 HAGR、DIST、集合操作、SPOOL、NTTS 以及 HTAB 操作符的数据总缓存大小，以 MB 为单位。有效值为 2～500000
HAGR_BLK_SIZE	1	动态，会话级	HAGR、DIST、集合操作、SPOOL、NTTS 以及 HTAB 操作符每次分配缓存（BLK）大小，以 MB 为单位，必须小于 HAGR_BUF_SIZE。有效值为 1～50

（续表）

参 数 名	默认值	属性	说　　明
MTAB_MEM_SIZE	8	静态	MTAB 缓存 BDTA 占用内存空间的大小，以 MB 为单位（1～1048576）
DICT_BUF_SIZE	5	静态	字典缓冲区大小，以 MB 为单位，有效值为 1～4096
HFS_CACHE_SIZE	160	动态，系统级	HFSI/U/D 时 HDTA_BUFFER 缓存池大小，单位为 MB。有效值为 160～4294967294
VM_STACK_SIZE	128	静态	系统执行时虚拟机堆栈大小，单位为 KB，堆栈的空间是从操作系统中申请的，有效值为 64～1024×1024
VM_POOL_SIZE	64	静态	系统执行时虚拟机内存池大小，在执行过程中用到的内存大部分是从这里申请的，它的空间是从操作系统中直接申请的，有效值为 32～1024×1024
VM_MEM_HEAP	0	静态	VM 是否使用 HEAP 分配内存。1 表示是，0 表示否
RFIL_RECV_BUF_SIZE	16	静态	控制服务器启动时，进行 REDO 操作过程中，REDO 日志文件恢复时 BUFFER 的大小，以 MB 为单位，有效值为 16～4000
N_MEM_POOLS	4	静态	内存池的数量，有效值为 1～128
COLDATA_POOL_SIZE	0	静态	BDTA 使用的共享内存池的大小，以 MB 为单位，有效值为 0～100，0 表示不启用 COLDATA_POOL
SSD_BUF_SIZE	0	静态	指定 SSD 缓冲区大小，以 MB 为单位。取值范围：0～4294967294，0 表示关闭
SSD_FILE_PATH	需要时指定	静态	SSD 缓冲区文件所在的文件夹路径，管理员保证其在 SSD 分区上
SSD_REF_BUF_SIZE	80	静态	SSD 缓冲专用 BUF 大小，SSD_BUF_ SIZE 不为 0 时有效。以 MB 为单位，取值为 20～4096
SSD_FLUSH_INTERVAL	10	静态	SSD 缓冲刷盘轮询间隔，以 ms 为单位，取值为 0～1000
SSD_FLUSH_STEPS	100	静态	SSD 缓存刷盘向前页数，取值为 10～100000

3. 线程参数

线程常见参数及含义见表 A-3。

表 A-3　线程常见参数及含义

参 数 名	默认值	属性	说　　明
WORKER_THREADS	4	静态	工作线程的数目，有效值为 1～64
TASK_THREADS	4	静态	任务线程个数（1～1000）
UTHR_FLAG	0	手动	用户线程标记，1 表示启用；0 表示不启用。 启用用户线程时，并行查询失效，并行查询的相关参数不起作用
FAST_RW_LOCK	1	手动	快速读写锁标记，1 表示启用，0 表示不启用
SPIN_TIME	4000	静态	线程在不能进入临界区时自旋的次数，有效值为 0～4000

（续表）

参 数 名	默认值	属性	说　明
WORK_THRD_STACK_SIZE	1024	静态	工作线程堆栈大小，以 KB 为单位。有效值为 64～4096
WORKER_CPU_PERCENT	0	手动	工作线程占 CPU 的比例，仅非 Windows 下有效（0～100）

4. 查询参数

查询常见参数及含义见表 A-4。

表 A-4　查询常见参数及含义

参 数 名	默认值	属 性	说　明
USE_PLN_POOL	1	静态	是否重用执行计划。0 表示禁止执行计划的重用；1 表示启用执行计划的重用功能；2 表示进行常量参数化优化
RS_CAN_CACHE	0	静态	是否启用结果集缓存，0 表示不启用；1 表示启用
RESULT_SET_LIMIT	10000	动态，会话级	一次请求可以生成的结果集最大个数。有效值为 1～65000
RESULT_SET_FOR_QUERY	0	动态，会话级	是否不生成非查询结果集。0 表示生成；1 表示不生成
SESSION_RESULT_SET_LIMIT	0	动态，系统级	会话上结果集个数上限，0 代表不限制（0～65000）
BUILD_FORWARD_RS	0	静态	游标是否生成结果集。0 代表生成；1 代表不生成
MAX_OPT_N_TABLES	8	动态，会话级	能优化的最大表连接个数。有效值为 3～8
CNNTB_MAX_LEVEL	20000	动态，会话级	层次查询的最大支持层次。有效值为 1～100000
BATCH_PARAM_OPT	1	静态	是否启用批量参数优化，0 表示禁用；1 表示启用，默认启用。当置为 1 时，不返回操作影响的行数
SESS_PLN_NUM	0	静态	每个会话上最多拥有的临时计划缓存个数，如果大于该个数，则会淘汰掉一部分缓存计划。0 表示不启用，也可以理解为会话上不会临时缓存计划。有效值为 0～4294967294。只有在计划重用参数 USE_PLN_POOL 启用时，该参数才会生效
CLT_CONST_TO_PARAM	0	静态	是否进行语句的常量参数化优化，0 表示不进行；1 表示进行
LIKE_OPT_FLAG	1	动态，会话级	LIKE 查询的优化开关，是否转换为 POSITION 处理。1 表示转换；0 表示不转换。取值为 0、1
ENABLE_DIST_IN_SUBQUERY_OPT	0	动态，系统级	优化子查询里面的 IN，0 表示不优化；1 表示优化
MAX_OPT_N_OR_BEXPS	7	动态，会话级	能参与优化的最大 OR 分支个数，超过时布尔表达式仅用于过滤使用。有效值为 7～16

（续表）

参 数 名	默认值	属　性	说　　明
USE_HAGR_FLAG	0	动态，会话级	当带有 DISTINCT 的集函数不能使用 SAGR 操作符时，是否使用 HAGR，1 表示使用；0 表示不使用
VIEW_PULLUP_FLAG	0	动态，会话级	是否对视图进行上拉优化，把视图转换为其原始定义，消除视图。0 表示不启用；1 表示启用
VIEW_PULLUP_MAX_TAB	7	动态，会话级	对视图进行上拉优化支持的表的个数。有效值为 6～16
HAGR_PARALLEL_OPT_FLAG	0	动态，会话级	MPP、并行下对 GROUPBY 的优化开关。1 表示优化；0 表示不优化。取值范围：0、1
REFED_EXISTS_OPT_FLAG	0	动态，会话级	值为 0 或 1，即 TURE 或 FALSE。是否将相关 EXISTS 优化为非相关 IN 查询：1 表示优化打开，0 表示优化关闭
CNNTB_HASH_TABLE_SIZE	100	动态，会话级	指定 CNNTB 操作符中创建 HASH 表的大小。取值为 100～100000000
OUTER_JOIN_INDEX_OPT_FLAG	0	动态，会话级	外连接优化为索引连接的优化开关。1 表示优化；0 表示不优化。取值为：0、1
MPP_HASH_LR_RATE	10	动态，会话级	MPP 下，对 HASHJOIN 节点，可以根据左右儿子代价的比值，调整 HASH_JOIN 的左右儿子的 MOTION 添加，从而影响计划。如果 CARD 比值超过此值，则小数据量的一方全部收集到主 EP 中来做。取值范围：10～10000
LPQ_HASH_LR_RATE	30	动态，会话级	LPQ 下，对 HASHJOIN 节点，可以根据左右儿子代价的比值，调整 HASH_JOIN 的左右儿子的 MOTION 添加，从而影响计划。 如果 CARD 比值超过此值，则小数据量的一方全部收集到主 EP 中来做。取值范围：10～10000
SEL_ITEM_HTAB_FLAG	0	动态，会话级	当查询项中有相关子查询时，是否做 HTAB 优化。0 表示不优化；1 表示优化
OR_CVT_HTAB_FLAG	1	动态，会话级	当查询条件 OR 中含有公共因子时，是否允许使用 HTAB 来进行优化。1 表示使用；0 表示不使用。取值范围：0、1
OP_SUBQ_CVT_IN_FLAG	1	动态，会话级	当查询条件为=(SUBQUERY)时，是否考虑转换为等价的 IN(SUBQUERY)。1 表示转换；0 表示不转换。取值范围：0、1
OPT_OR_FOR_HUGE_TABLE_FLAG	1	动态，会话级	是否使用 HFSEK 优化 HUGE 表中列的 OR 过滤条件。1 表示使用；0 表示不使用。取值范围：0、1

5. 检查点参数

检查点常见参数及含义见表 A-5。

表 A-5 检查点常见参数及含义

参 数 名	默认值	属性	说 明
CKPT_RLOG_SIZE	100	静态	产生多大日志文件后做检查点，以 MB 为单位。有效值范围：0～4294967294
CKPT_DIRTY_PAGES	10000	静态	产生多少脏页后产生检查点，以页为单位。有效值范围：0～4294967294
CKPT_INTERVAL	1800	静态	指定检查点的时间间隔。以秒为单位，为 0 时表示不自动定时做检查点。有效值范围：0～4294967294
CKPT_FLUSH_RATE	5	静态	检查点刷盘比例。有效值范围：1～100
CKPT_FLUSH_PAGES	1000	静态	检查点刷盘的最小页数。有效值范围：1000～100000
FORCE_FLUSH_PAGES	10	静态	调度线程启动刷脏页流程时，每个 BUFFERPOOL 写入磁盘的脏页数。有效值范围：0～1000

6. I/O 参数

I/O 常见参数及含义见表 A-6。

表 A-6 I/O 常见参数及含义

参 数 名	默认值	属性	说 明
DIRECT_IO	0	静态	非 Windows 下有效，0 表示使用文件系统缓存；1、2 表示不使用文件系统缓存
IO_THR_GROUPS	2	静态	非 Windows 下有效，表示 IO 线程组个数。有效值范围：0～512
FIL_CHECK_INTERVAL	0	动态，系统级	指定检查数据文件是否存在的时间间隔，单位为秒。有效值范围：0～4294967294。0 表示不检查

7. 数据库相关参数

数据库常见参数及含义见表 A-7。

表 A-7 数据库常见参数及含义

参 数 名	默认值	属性	说 明
MAX_SESSIONS	100	静态	系统允许同时连接的最大数，同时受到 LICENSE 的限制，取二者中较小的值，有效值范围：1～30000
MAX_SESSION_STATEMENT	100	静态	单个会话上允许同时打开的语句句柄最大数，有效值范围：64～20480

（续表）

参 数 名	默认值	属性	说　　明
MAX_CONCURRENT_OLAP_QUERY	0	静态	OLAP 模式下能同时执行的复杂查询个数，超过限制后，复杂查询将在执行阶段进入等待。0 表示不做限制。有效值范围为 0～100。注：此处的复杂查询是指计划中存在 CARD 超过一千万的节点，含有哈希连接、哈希分组等耗内存操作符，且不涉及系统表
MAX_EP_SITES	64	手动	MPP 环境下 EP 站点的最大数量，有效值范围：2～1024
PORT_NUM	5236	静态	服务器通信端口号，有效值范围：1024～65534
CHECK_DB_IS_ACTIVE	1	静态	设置为 1，则自动检查数据库实例是否处于运行状态，以保证一个数据实例不被多次的实例化；设置为 0，则靠用户自己保证。 注：如果设置为 0，会导致客户端工具 MANAGER 登录界面中的数据库实例浏览信息不准确
DDL_AUTO_COMMIT	1	动态，会话级	指定 DDL 语句是否自动提交，1 表示自动；0 表示手动。当 COMPATIBLE_MODE=1 时，DDL_AUTO_COMMIT 的实际值均为 0
COMPRESS_MODE	0	动态，会话级	建表时是否默认压缩。0 表示不进行；1 表示进行
PK_WITH_CLUSTER	0	动态，会话级	建立主关键字时，是否默认指定为 CLUSTER，0 表示不指定；1 表示指定
SP_REBUILD	0	静态	系统启动时是否重建存储过程，0 表示不重建；1 表示重建
EXPR_N_LEVEL	200	动态，会话级	表达式最大嵌套层数。有效值范围：10～1000
BDTA_SIZE	1000	静态	BDTA 缓存的记录数。有效值范围：1～10000
OLAP_FLAG	0	动态，会话级	启用联机分析处理，0 表示不启用；1 表示启用
JOIN_HASH_SIZE	500000	动态，会话级	HASHJOIN 操作时，HASH 表的大小，以 CELL 为单位。有效值范围：0～100000000
USE_DHASH_FLAG	0	动态，会话级	是否使用动态 HASH。0 表示不使用；1 表示使用动态 HASH1；2 表示使用动态 HASH2
FILES_OPENED	60	静态	最大打开文件数。有效值：60～100
FAST_COMMIT	0	静态	批量提交事务的个数（0～100）
ISO_IGNORE	0	动态，会话级	是否忽略显示设置的事务隔离级别，1 表示忽略，0 表示不忽略
TEMP_SIZE	10	静态	临时数据库大小，以 MB 为单位。有效值范围：10～1048576
FILE_TRACE	0	静态	日志中是否记录文件操作，0 表示不记录；1 表示记录
CACHE_POOL_SIZE	10	静态	SQL 缓冲池大小，以 MB 为单位。有效值范围：1～10240
STAT_COLLECT_SIZE	10000	动态，会话级	统计信息收集时，样本的最小行数。有效值范围：0～10000000

（续表）

参 数 名	默认值	属性	说　明
PHC_MODE_ENFORCE	0	动态，会话级	允许使用的 JOIN 实现方式，0 表示所有；1 表示允许 NESTLOOP；2 表示允许 INDEXJOIN；4 表示允许 HASHJOIN；8 表示允许 MERGEJOIN 使用上述值或运算的结果，如值 11=1+2+8 表示可以使用除 HASHJOIN 外所有的连接方式。任何情况下 NESTLOOP 始终被优化器考虑
MAX_PARALLEL_DEGREE	1	动态，会话级	用来设置最大并行任务个数。取值范围：1～128。默认值为 1，表示无并行任务。当 PARALLEL_POLICY 值为 1 或 2 时该参数值才有效
PARALLEL_POLICY	0	静态	用来设置并行策略。取值范围：0、1 和 2，默认为 0。其中，0 表示不支持并行；1 表示自动配置并行工作线程个数（与物理 CPU 核数相同）；2 表示手动设置并行工作线程数
PARALLEL_THRD_NUM	10	静态	用来设置并行工作线程个数，仅当 PARALLEL_POLICY 值为 2 时才启用此参数。有效值范围：1～1024
ENABLE_MS_PUSH_DOWN	0	动态，系统级	在多表连接中是否允许强制采用归并连接实现子查询，1 表示允许；0 表示不允许
UPD_TAB_INFO	0	静态	系统启动时是否更新表信息（如行数），1 表示启用更新；0 表示不更新
ENABLE_IN_VALUE_LIST_OPT	0	动态，系统级	是否允许用索引连接的方式进行 IN 常量列表的过滤，1 表示允许；0 表示不允许
ENABLE_DIST_VIEW_UPDATE	0	动态，系统级	是否支持更新含有 DISTINCT 的视图，1 表示允许；0 表示不允许，当 COMPATIBLE_MODE=1 时，ENABLE_DIST_VIEW_UPDATE 的实际值均为 1
ENABLE_QUERY_EXP_OPT	0	动态，系统级	是否允许相关查询表达式优化，1 表示允许；0 表示不允许
STAR_TRANSFORMATION_ENABLED	0	动态，会话级	是否允许对星形模型查询改写以使用位图连接索引。0 表示不启用；1 表示启用
FIRST_ROWS	100	静态	结果集一个消息中返回的最大行数（1～1000）
LIST_TABLE	0	动态，会话级	默认情况下，创建的表是否为 LIST 表，0 表示否；1 表示是
NEW_FILE_POLICY	0	手动	备份数据库在创建文件时，使用的路径策略，0 为相对路径，1 为绝对路径。配置为 1 时，需要确保主备机的软硬件环境完全一致
ENABLE_SPACELIMIT_CHECK	1	静态	是否启用检查 SPACELIMIT。1 表示启用；0 表示不启用
BUILD_VERTICAL_PK_BTREE	0	手动	创建 VERTICAL 表的时候，是否创建主键。1 表示创建，0 表示不创建
BDTA_PACKAGE_COMPRESS	0	动态，会话级	是否启动 BDTA 压缩传递功能，1 表示压缩；0 表示不压缩

8. 预先装载表参数

预先装载表常见参数及含义见表 A-8。

表 A-8　预先装载表常见参数及含义

参 数 名	默认值	属性	说　　明
LOAD_TABLE	空串	手动	在服务器启动时预先装载的表的完整表名，即“模式名.表名”，多个表之间用逗号分隔，最多可指定 10 个表
CLT_CACHE_TABLES	空串	手动	指定可以在客户端缓存的表。表名必须带模式名前缀，如果表名或模式名中包含特殊字符，需要使用双引号包含。如果指定多个缓存表，则必须以逗号间隔。服务器最多支持指定 100 个可缓存表。为避免参数值太长导致 INI 文件分析困难，允许在 INI 文件中设置多行 CLT_CACHE_TABLES 参数

9. 日志参数

日志常见参数及含义见表 A-9 所示。

表 A-9　日志常见参数及含义

参 数 名	默认值	属性	说　　明
LOG_BUF_SIZE	512	静态	单个日志缓冲区大小（以日志页个数为单位），取值只能为 2 的次幂值，最小值为 1，最大值为 20480
LOG_POOL_SIZE	128	静态	最大日志缓冲区大小（以 MB 为单位）。有效值范围：16～1024
LOG_REDO_THREAD_NUM	2	静态	日志线程数（1～10）
RLOG_PARALLEL_ENABLE	0	静态	是否启动并行日志，1 表示启用；0 表示不启用
RLOG_RESERVE_SIZE	2048	静态	INSERT/DELETE/UPDATE 等操作预留的日志空间大小（以日志页个数为单位）。有效值范围：0～131072

10. 事务参数

事务常见参数及含义见表 A-10。

表 A-10　事务常见参数及含义

参 数 名	默认值	属性	说　　明
ISOLATION_LEVEL	1	静态	系统默认隔离级别。1 表示读提交；3 表示可串行化
DDL_WAIT_TIME	10	动态，会话级	DDL 操作的锁超时时间，以秒为单位。有效值范围 0～60
FAST_RELEASE_SLOCK	1	动态，系统级	是否启用快速释放 S 锁，1 表示启用；0 表示不启用

（续表）

参 数 名	默认值	属性	说　明
SESS_CHECK_INTERVAL	3	动态，会话级	循环检测会话状态的时间间隔，以秒为单位。有效值范围：1～60
LOCK_TIMEOUT	10	动态，系统级	在 UTHR_FLAG=1 的模式下，锁等待时间，以秒为单位，有效值为 1～4294967294
SERIALIZABLE_CHECK_INTERVAL	300	动态，系统级	当串行化事务在指定时间内没有提交时，强制断开连接，以秒为单位。有效值范围：60～3600
READ_COMMITED_CHECK_INTERVAL	900	动态，系统级	读提交事务且生成了回滚记录情况下，强制断开的检测时间。有效值范围：60～3600
DML_ONLY	0	动态，系统级仅在服务器为 MOUNT 状态时才能修改	0 表示操作没有任何限制；1 表示 DDLFORBIDDEN，支持所有 DML 操作（包括 VERTICAL/HUGE 表），支持 FASTLOADER，不支持 DDL 操作；2 表示 DMLONLY，仅支持普通表的 DML 操作，不支持 DDL 操作，不支持 FASTLOADER，不支持 VERTICAL/HUGE 表的 DML 操作
NOWAIT_WHEN_UNIQUE_CONFLICT	0	静态	插入数据时，如果和未提交数据有 UNIQUE 约束的冲突，是否等待未提交事务结束，0 表示等待，直至未提交事务结束；1 表示不等待，立即返回错误
UNDO_EXTNET_NUM	16	静态	初始回滚簇数目，有效值范围：1～4294967294
MAX_DE_TIMEOUT	10	动态，会话级	C 外部函数的执行超时时间，以秒为单位。有效值范围：1～3600
TRANSACTIONS	75	静态	指定一个会话中可以并发的自治事务数量。取值范围：1～1000
MVCC_RETRY_TIMES	5	静态	指定发生 MVCC 冲突时的最大重试次数。有效值范围：1～4294967294
ENABLE_FLASHBACK	0	动态，系统级	是否启用闪回查询，0 表示不启用；1 表示启用
UNDO_RETENTION	900	动态，系统级	事务提交后回滚页保持时间，单位为秒。有效值范围：1～86400
PURGE_DEL_OPT	0	动态，系统级	PURGEDELETE 操作是否进行优化。0 表示不优化，1 表示优化

11. 安全参数

安全常见参数及含义见表 A-11。

表 A-11　安全常见参数及含义

参 数 名	默认值	属性	说　　明
PWD_POLICY	2	动态，系统级	设置系统默认口令策略。0 表示无策略；1 表示禁止与用户名相同；2 表示口令长度不小于 6；4 表示至少包含一个大写字母（A～Z）；8 表示至少包含一个数字（0～9）；16 表示至少包含一个标点符号（英文输入法状态下，除“和空格外的所有符号；若为其他数字，则表示配置值的和，如 3＝1+2，表示同时启用第 1 项和第 2 项策略。当 COMPATIBLE_MODE=1 时，PWD_POLICY 的实际值均为 0
ENABLE_ENCRYPT	0	静态	通信所采用的加密类型。0 表示不加密；1 表示 SSL 加密
ENABLE_AUDIT	0	动态，系统级	审计开关，0 表示关闭；1 表示打开普通审计；2 打开普通审计和实时审计
AUDIT_FILE_FULL_MODE	1	静态	审计文件满后的处理方式，1 表示删除文件；2 表示不删除文件，也不添加审计记录
AUDIT_MAX_FILE_SIZE	100	动态，系统级	审计文件的最大大小，以 MB 为单位。有效值范围：1～4096
ENABLE_OBJ_REUSE	0	静态	是否支持客体重用，0 表示不支持；1 表示支持。该参数设置仅安全版有效
ENABLE_REMOTE_OSAUTH	0	静态	是否支持远程操作系统认证，0 表示不支持；1 表示支持。该参数设置仅安全版有效
MSG_COMPRESS_TYPE	1	静态	是否加载压缩动态库，0 表示不加载；1 表示加载
ENABLE_STRICT_CHECK	0	静态	是否检查存储过程中 EXECUTEIMMEDIATE 语句的权限，1 表示表示检查；0 表示不检查
MAC_LABEL_OPTION	1	动态，系统级	用于控制 SP_MAC_LABEL_FROM_CHAR 过程的使用范围。 0 表示只有 SSO 可以调用； 1 表示所有用户都可以调用； 2 表示所有用户可以调用，但是非 SSO 用户不会主动创建新的 LABEL
RESTRICT_DBA	0	手动	限制 DBA 的 ANY 权限。1 表示限制；0 表示不限制。只有 SYSSSO 有此权限
LDAP_HOST	空串	手动	LDAP 服务器 IP 地址

12. 兼容性参数

兼容性常见参数及含义见表 A-12。

表 A-12 兼容性常见参数及含义

参 数 名	默认值	属性	说 明
BACKSLASH_ESCAPE	0	动态，会话级	语法分析对字符串中的反斜杠是否需要进行转义处理，0 表示不进行转义处理；1 表示进行转义处理
STR_LIKE_IGNORE_MATCH_END_SPACE	1	动态，会话级	LIKE 运算中是否忽略匹配串的结尾 0。0 表示不忽略；1 表示忽略
CLOB_LIKE_MAX_LEN	32	静态	LIKE 语句中 CLOB 类型的最大长度，单位为 KB，有效值范围：8～102400
EXCLUDE_DB_NAME	空串	静态	服务器可以忽略的数据库名列表，各数据库名以逗号“,”隔开，数据库名不需要加引号
MS_PARSE_PERMIT	0	静态	是否支持 SQL Server 的语法。0 表示不支持；1 表示支持。当 COMPATIBLE_MODE=3 时，MS_PARSE_PERMIT 的实际值为 1
COMPATIBLE_MODE	0	静态	是否兼容其他数据库模式。0 表示不兼容，1 表示兼容 SQL92 标准，2 表示兼容 Oracle，3 表示兼容 SQL Server，4 表示兼容 MySQL
DROP_CASCADE_VIEW	0	动态，会话级	删除表或者视图的时候是否级联删除视图，0 表示只删除表或者视图；1 表示删除表或者视图时删除关联的视图，当 COMPATIBLE_MODE=1 时，DROP_CASCADE_VIEW 的实际值为 1

13. 跟踪监控参数

跟踪监控常见参数及含义见表 A-13、表 A-14。

表 A-13 跟踪监控相关参数及含义

参 数 名	默认值	属性	说 明
SQL_TRACE_MASK	1	动态，系统级	LOG 记录的语句类型掩码，是一个格式化的字符串，表示一个 32 位整数上哪一位将被置为 1，置为 1 的位表示该类型的语句要记录，格式如下：位号:位号:位号。例如，3:5:7 表示第 3、5、7 位上的值被置为 1。每一位的含义见表 A-14
SVR_LOG_FILE_NUM	0	动态，系统级	总共记录多少个日志文件，当日志文件达到这个设定值以后，再生成新的文件时，会删除最早的那个日志文件，日志文件的命令格式为 LOG_COMMIT_时间.LOG。当这个参数配置成 0 时，按传统的日志文件记录，也就是 LOG_COMMIT01.LOG 和 LOG_COMMIT02.LOG 相互切换着记录。有效值范围：0～4294967294

（续表）

参 数 名	默认值	属性	说 明
SVR_LOG	0	动态，系统级	是否打开 SQL 日志功能，0 表示关闭；1 表示日志文件为非切换模式，但输出的日志格式是详细模式；2 表示日志文件为切换模式，输出的日志也是详细模式；3 表示日志为非切换模式，但输出日志为简单模式
SVR_LOG_SWITCH_COUNT	500	动态，系统级	一个日志文件中的 SQL 记录条数达到多少条之后系统会自动将日志切换到另一个文件中。有效值范围：1000～1000000
TRACE_PATH	DM.INI 文件所在目录	手动	存放系统 TRACE 文件的路径

表 A-14 SQL_TRACE_MASK 各位参数的含义

位 数	含 义	位 数	含 义
1	全部记录（全部记录并不包含原始语句）	16	触发器对象操作（TRIGGERDDL）
2	全部 DML 类型语句	17	序列对象操作（SEQUENDDL）
3	全部 DDL 类型语句	18	模式对象操作（SCHEMADDL）
4	UPDATE 类型语句（更新）	19	库对象操作（DATABAEDDL）
5	DELETE 类型语句（删除）	20	用户对象操作（USERDDL）
6	INSERT 类型语句（插入）	21	索引对象操作（INDEXDDL）
7	SELECT 类型语句（查询）	22	绑定参数
8	COMMIT 类型语句（提交）	23	存在错误的语句(语法错误、语义分析错误等)
9	ROLLBACK 类型语句（回滚）	24	是否需要记录执行语句
10	CALL 类型语句（过程调用）	25	是否需要打印计划和语句执行的时间
11	BACKUP 类型语句（备份）	26	是否需要记录执行语句的时间
12	RESTORE 类型语句（恢复）	27	原始语句（服务器从客户端收到的未加分析的语句）
13	表对象操作（TABLEDDL）	28	是否记录参数信息，包括参数的序号、数据类型和值
14	视图对象操作（VIEWDDL）	29	是否记录事务相关事件
15	过程或函数对象操作(PROCEDUREDDL)		

附录 B　达梦数据库技术支持

如果在安装或使用达梦数据库系统及其相应产品时出现了问题，请首先访问 Web 站点 http://www.dameng.com/。在此站点我们收集整理了安装使用过程中一些常见问题的解决办法，相信会对用户有所帮助。

用户也可以通过以下途径与达梦数据库有限公司联系，达梦数据库有限公司技术支持工程师会为用户提供服务。

达梦数据库（武汉）有限公司
地址：武汉市关山一路特 1 号光谷软件园 C6 栋 5 层
邮编：430073
电话：（+86）027-87588000
传真：（+86）027-87588000-8039

达梦数据库（北京）有限公司
地址：北京市海淀区北三环西路 48 号数码大厦 B 座 905
邮编：100086
电话：（+86）010-51727900
传真：（+86）010-51727983

达梦数据库（上海）有限公司
地址：上海市闸北区江场三路 28 号 301 室
邮编：200436
电话：（+86）021-33932716
传真：（+86）021-33932718
地址：上海市浦东张江高科技园区博霞路 50 号 403 室
邮编：201203
电话：（+86）021-33932717
传真：（+86）021-33932717-801

达梦数据库（广州）有限公司

地址：广州市荔湾区中山七路 330 号荔湾留学生科技园 703 室

邮编：510145

电话：（+86）020-38371832

传真：（+86）020-38371832

达梦数据库（海南）有限公司

地址：海南省海口市玉沙路富豪花园 B 座 1602 室

邮编：570125

电话：（+86）0898-68533029

传真：（+86）0898-68531910

达梦数据库（南宁）办事处

地址：广西壮族自治区南宁市科园东 5 路 4 号南宁软件园 5 楼

邮编：530003

电话：（+86）0771-2184078

传真：（+86）0771-2184080

达梦数据库（合肥）办事处

地址：合肥市包河区马鞍山路金帝国际城 7 栋 3 单元 706 室

邮编：230022

电话：（+86）0551-3711086

达梦数据库（深圳）办事处

地址：深圳市福田区皇岗路高科利大厦 A 栋 24E

邮编：518033

电话：（+86）0755-83658909

传真：（+86）0755-83658909

技术服务

电话：400-991-6599

邮箱：dmtech@dameng.com